José Emanuel da Costa Cunha

Licenciado em Engenharia Informática

Deteção automática de atividades agronómicas

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Orientadores : Carlos Viegas Damásio, Prof. Associado, Universidade Nova de Lisboa
Armanda Rodrigues, Prof. Auxiliar, Universidade Nova de Lisboa

Júri:

Presidente:	Doutor Rodrigo Seromenho Miragaia Rodrigues
Arguente:	Doutor José Carlos dos Santos Danado
Vogal:	Doutor Carlos Augusto Isaac Piló Viegas Damásio

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2013

Deteção automática de atividades agronómicas

Copyright © José Emanuel da Costa Cunha, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

À minha família.

Agradecimentos

Em primeiro lugar, quero agradecer aos meus orientadores, Professor Doutro Carlos Viegas Damásio e à Professora Doutora Maria Armanda Grueau por serem incansáveis e por estarem sempre disponíveis para me ajudar e orientar. Agradecer pela ajuda que me deram nos momentos de discussão, na leitura pormenorizada e críticas construtivas que, de diversas formas, permitiram a construção desta dissertação. Quero agradecer pelo estímulo que sempre me deram ao longo do desenvolvimento desta dissertação e sem o qual a conclusão desta dissertação não seria possível.

Em segundo lugar quero agradecer à minha família por todo apoio que me deram e pela paciência que tiveram comigo nos momentos mais complicados. Em especial à minha mãe, ao meu pai, à minha avó e à minha irmã Catarina. Sem vocês não seria possível.

Por último, um muito obrigado aos meus amigos, especialmente ao grupo "Sala 236" e "Los Mariachis". Sem a vossa compreensão e ajuda, nos momentos em que era preciso rir ou tomar decisões, teria sido mais complicado o atingir desta etapa.

A todos,

MUITO OBRIGADO.

Resumo

A partir da informação obtida pelos vários sensores dos *smartphones*, como o *GPS* ou o acelerómetro, é possível construir aplicações capazes de captar informação sobre o contexto espacial em que estamos, surgindo a possibilidade de automatizar muitas das atividades executadas pelos utilizadores. Considerando ainda os avanços nos Sistemas de Informação Geográfica, a aplicação de todos estes conceitos à chamada agricultura moderna, traz uma oportunidade de aumentar o nível de produtividade de um técnico, na realização das suas atividades do dia-a-dia. Nesta dissertação foi desenvolvido um sistema que, com o mínimo de intervenção humana, permite detetar de forma automática algumas das atividades do técnico, bem como os lugares visitados pelo mesmo durante o seu dia de trabalho, tirando partido de uma captação *GPS* recolhida por ele.

Em primeiro lugar, é feita uma deteção dos locais importantes visitados pelo técnico, chamados pontos de estadia. Esses locais são depois agrupados através de uma técnica de *clustering* (agrupamento) e no final toda a informação é complementada com moradas e estabelecimentos reais provenientes de uma pesquisa a um serviço de *geocoding*. Após a extração dos lugares, são identificadas as atividades efetuadas pelo técnico. Algumas são extraídas diretamente por estarem associadas a lugares que foram identificados previamente, enquanto que outras são identificadas através de algoritmos desenvolvidos especificamente para a dissertação. Os algoritmos têm por base uma análise ao tipo de movimento realizado entre os pontos de estadia e a deteção de alguns padrões, sendo no final feita a extracção das atividades associadas ao dia-a-dia do técnico. O sistema foi complementado com uma interface *web* associada à aplicação desenvolvida, que permite visualizar toda a informação extraída.

Palavras-chave: Dispositivos móveis, registos *GPS*, extração de locais, inferência de atividades, extração de padrões.

Abstract

From the information obtained by the multiple sensors of *smartphones*, such as the GPS or accelerometer, applications can be built to capture information about the spatial context in which users are, enabling the possibility to automate many of the manual tasks performed by them. Thus, with advances in the field of Geographic Information Systems, when applying these concepts to what is called modern agriculture, the opportunity arises for an agronomist to increase his level of productivity in his day-to-day activities. For this thesis, a system to automatically detect various activities performed by an agronomist, as well as the locations he has been to, was developed. This system requires minimum human intervention and it is based on a GPS recording file collected by the user.

Firstly, a detection of the sites important for the technician is made, called *Staypoints*. These sites are then clustered, using a clustering technique described in the literature, and then consolidated with information obtained through *geocoding* services. After the place extraction and identification, the activities performed by the technician are identified. Some are inferred directly based on the places that were previously identified, whereas other activities are identified using algorithms specifically developed for this thesis. The algorithms are based on the type of motion performed between the Staypoints and the detection of some patterns. In the end all the technician day-to-day activities are extracted. The system was complemented with a web interface associated with the developed application, enabling the visualization of all the information extracted

Keywords: Mobile devices, *GPS* records, location recognition, activity inference, pattern recognition.

Conteúdo

Lista de Figuras

Listagens

1

Introdução

1.1 Motivação

Os avanços registados ao longo dos últimos anos na área da tecnologia móvel, tornaram-na de tal forma banal que esta já faz parte, de forma intrínseca, da vida do dia-a-dia de cada um. A ubiquidade da tecnologia permite ao utilizador tirar o máximo partido do melhor que esta tem para oferecer. Grande exemplo disso são os *smartphones*. Devido aos vários sensores que estão presentes nestes dispositivos é possível obter vários tipos de informação a partir do meio que nos rodeia. Desde o GPS ao acelerómetro, todos os sensores podem contribuir de alguma forma para ajudar a facilitar a vida de quem os usa.

Dos vários tipos de *smartphones* existentes no mercado, cada um com o seu sistema operativo, um dos grandes causadores da massificação são os dispositivos *Android* que, pelo seu custo acessível, permitiram a adoção das massas a este tipo de *gadgets*. Devido a este fenómeno, o número de aplicações existentes para este tipo de equipamentos tem surgido a um ritmo impressionante. Ou seja, o contributo deste tipo de dispositivos torna-se uma mais valia, porque a automatização de muitos processos facilita grandemente a vida dos utilizadores.

Para além do desenvolvimento da tecnologia móvel, os grandes avanços das tecnologias na área dos Sistema de Informação Geográfica [18], disponibilizam uma forma de se conseguir observar e tratar dados espaciais de forma fácil e bastante intuitiva para o utilizador. Estes sistemas permitem a manipulação e tratamento de dados georreferenciados de forma pormenorizada, complementando a informação obtida, por exemplo, por *GPS*.

Consequência de todo este avanço tecnológico, é a introdução da tecnologia em várias áreas de trabalho, como por exemplo, na Agricultura [1], [20]. A introdução de mecanismos que permitam automatizar certas atividades praticadas neste meio, permite aumentar o nível de produtividade, bem como o retorno de se investir neste tipo de setor económico. A dissertação que foi desenvolvida e que será descrita neste documento, teve como ideia base as motivações apresentadas aproveitando assim a oportunidade de criar uma solução para o problema apresentado na área da agricultura detalhado na secção 1.2.

1.2 Descrição do problema

Uma exploração agrícola é uma unidade técnico-económica que utiliza fatores de produção comuns, tais como: mão de obra, máquinas, instalações, parcelas de terreno, entre outros, que deve estar localizada num local bem determinado e identificável [1]. Uma exploração agrícola pode ser subdividida em parcelas. Uma parcela, corresponde a uma área delimitada geograficamente, identificada em função da ocupação de solo e representa uma porção contínua de terreno, homogénea, com limites estáveis, agronómica e geograficamente [2]. Numa parcela, dependendo da exploração que se está a fazer nela, podem existir linhas/grelhas de plantações de árvores ou plantas. Em suma, uma exploração agrícola é divida em várias parcelas que podem ter ou não várias divisórias, as quais designamos por linhas ou grelhas. As distâncias entre as plantas na grelha são designadas compassos.

Para além das linhas de plantação existentes nas várias parcelas agrícolas, existem zonas especiais com determinadas propriedades. Uma parcela pode conter zonas que, pela infestação de alguma bactéria ou inseto, se tornaram inviáveis ou problemáticas para plantação. A estas áreas dá-se o nome de zonas afetadas. Numa grande parte dos insetos, os machos localizam as fêmeas da sua espécie através de substâncias especiais que se espalham no ar, designadas feromonas. Estes odores, correspondem a substâncias que são diferentes para cada uma das espécies em questão. Por forma a monitorizar e evitar as infestações nas árvores ou plantas, são montadas armadilhas com feromonas.

Qualquer cultura precisa de água, entre outros produtos, tais como adubos ou fertilizantes, para que se possa desenvolver adequadamente. Para a irrigação das culturas podem ser utilizados pivôs. Um pivô é uma estrutura que, ou girando em volta duma área circular ou percorrendo perpendicularmente toda a plantação, asperge água por toda a cultura.

Um técnico agrícola é um profissional que tem como objetivo supervisionar as atividades realizadas no campo e aconselhar sobre as mesmas. No entanto, para além de supervisionar e avaliar as várias agriculturas presentes numa exploração, um técnico está

[1] http://metaweb.ine.pt/sim/CONCEITOS/Detalhe.aspx?cnc_cod=657&cnc_ini=07-10-2008
[2] http://www.ifap.min-agricultura.pt/portal/page/portal/ifap_publico/GC_informacoes/GC_parcelario

também encarregue de falar com os clientes, fazer as marcações e as observações necessárias no terreno. Seja no campo, no escritório do cliente, ou na sede da empresa, o técnico deve reunir de modo a perceber o que é necessário fazer ou que tipo de ações são necessárias num determinado momento. Para além das reuniões, é esperado também que o técnico se dirija à exploração de modo a fazer os croquis e as observações necessárias nas várias parcelas.

Assim, um técnico agrícola segue uma vida que se pode considerar bastante variada, mas com um conjunto de atividades bem definidas. É importante, para um técnico, ter um registo do que fez durante o dia, quais os locais que visitou ou as marcações que realizou. Um dia de um técnico pode ser resumido às seguintes atividades principais:

- Conduzir;
- Comer (passar algum tempo num restaurante);
- Abastecer (estar numa bomba de gasolina);
- Visitar locais (por exemplo, ir ao banco, empresa...);
- Participar em reuniões (com os clientes, na empresa ou no campo, com os fornecedores);
- Visitar exploração, lá realizando diversas atividades:
  - Marcar parcelas;
  - Remover partes de parcelas;
  - Marcar zonas afetadas;
  - Marcar pivôs;
  - Marcar armadilhas;
  - Marcar linhas e grelhas;
  - Efetuar observações (armadilhas, estado das plantas).

Baseado nas descrições e motivações apresentadas, surge assim o problema para esta dissertação. O problema que se pretende resolver, prende-se com a necessidade de automatizar os processos manuais de registo e identificação das atividades mencionadas, das marcações nas parcelas e dos locais importantes para o técnico. Ou seja, a extração e o registo de toda a informação mencionada, deverá exigir o mínimo esforço humano, ou seja, apenas o necessário para a aquisição dos dados usando o *smartphone*. O problema provém da necessidade de desenvolver uma forma de facilitar a aquisição e o tratamento dos dados para extração de informação, com o mínimo de intervenção humana, de modo a aumentar a produtividade e a diminuir o tempo despendido pelos técnicos na elaboração deste tipo de registos. Por fim, o relatório de atividade diária do técnico deverá

ser gerado quase automaticamente. O relatório conterá para além dos lugares visitados, uma descrição sequencial das atividades apresentadas.

Uma alternativa à forma de marcação e desmarcação de parcelas poderia passar pela sua identificação direta a partir, por exemplo, do *Google Maps* [3]. No entanto, existe um problema que é o facto de este serviço não fornecer sempre as imagens mais atuais dos terrenos, o que poderia levar a um impasse na identificação das parcelas. Outro problema em usar este serviço está associado aos limites das parcelas e ao tipo de cultura presente. É frequente as parcelas variarem a sua área durante o ano e as culturas que contêm, pois os agricultores alugam partes de parcelas fazendo assim com que os limites destas se alterem frequentemente.

1.3 Objetivos

Esta dissertação tem como objetivo principal a criação de uma aplicação que, dado um registo *GPS* de um técnico agrícola, permita gerar um relatório que lhe possibilite visualizar quais as atividades que realizou durante a captação, os lugares que visitou e as marcações efetuadas, tal como explicado na secção 1.2.

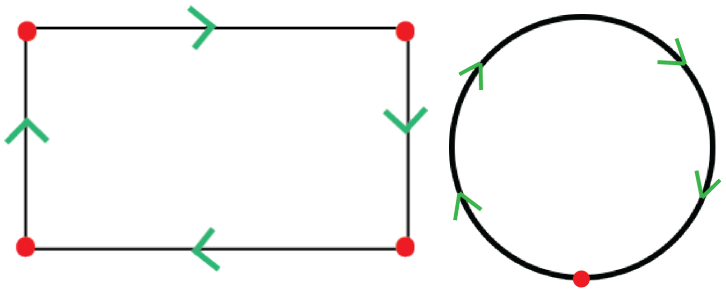
Para atingir o objetivo principal desta dissertação, foram contemplados um conjunto de sub-objetivos que permitiram a realização final do objetivo pretendido. Assim os sub-objetivos desta dissertação são:

1. Identificação de lugares;
2. Identificação de atividades:
   (a) As associadas ao dia-a-dia do técnico;
   (b) Marcar parcela;
   (c) Desmarcar parcela;
   (d) Pivôs;
3. Interface:
   (a) Visualização de resultados:
      i. Relatório Final;
   (b) Identificação/correção manual dos lugares;

Foram dadas indicações aos técnicos, que recolhem os dados, que sempre que quisessem fazer marcações ou observações nas parcelas, o deveriam fazer através da execução de padrões pré-definidos, por forma a que, posteriormente, as atividades em causa pudessem ser identificadas e extraídas através de métodos que identificassem os padrões nos registos submetidos. Para fazerem a marcação de parcelas, foi-lhes indicado que as

[3] https://maps.google.com

deveriam circundar sempre no sentido dos ponteiros do relógio, devendo começar e acabar no mesmo vértice da parcela. A situação está definida nas figuras 1.1(a) e 1.1(b). Por outro lado, caso quisessem excluir parte de uma parcela, ou seja, desmarcá-la, a indicação foi similar, mas com a diferença que, a zona a excluir deveria ser circundada no sentido contrário ao dos ponteiros do relógio.



(a) Indicações para marcação de parcelas regulares

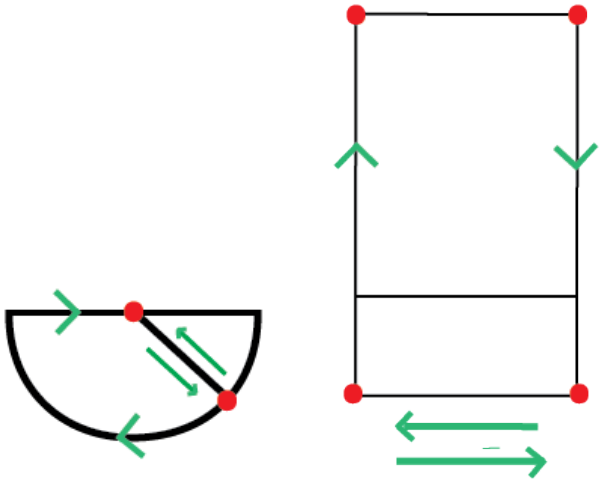
(b) Indicações para marcação de parcelas circulares

Figura 1.1: Tipos de marcações

Foram dadas também indicações relacionadas com a marcação dos pivôs. A marcação de pivôs deveria ser feita da seguinte forma:

Parcela circular: Caso fosse uma parcela circular o pivô deveria ser do tipo circular. Para o identificar, depois da marcação da parcela, o técnico deveria deslocar-se até à base do pivô parando algum tempo no centro de rotação. Depois de estar parado nesse local durante algum tempo, o técnico deveria voltar ao local onde terminou a marcação da parcela pelo mesmo caminho que tinha feito anteriormente. Esta marcação encontra-se representada na figura 1.2(a).

Parcela retangular/quadrada: No caso de ser uma parcela retangular/quadrada, o pivô percorreria a parcela de forma perpendicular. Desta forma, para o identificar, depois de marcar a parcela, o técnico deveria dirigir-se à ponta do pivô, que vai estar assente num vértice que delimita a parcela. Depois de parar durante algum tempo no vértice, deveria dirigir-se à outra ponta do pivô, parar durante algum tempo e, por fim, voltar ao vértice inicial. É possível observar o padrão descrito na figura 1.2(b).



(a) Indicações para marcação de pivôs circulares

(b) Indicações para marcação de pivôs horizontais

Figura 1.2: Marcações de pivôs

1.4 Contribuições

Com o desenvolvimento desta dissertação, pretendeu-se trazer uma contribuição relevante na área da identificação de atividades de cariz espacial, mais propriamente na identificação das atividades relacionadas com a atividade "Visitar exploração", apresentada em cima,tendo por base um registo de um *GPS*. Foram desenvolvidos mecanismos capazes de identificar o padrão que caracteriza este tipo de atividades no registo. Apesar das contribuições estarem, no caso desta dissertação, mais direcionadas à agricultura, os mecanismos a desenvolver poderão servir para a marcação de outros tipos de áreas, como por exemplo a identificação dos vários prédios numa urbanização.

Todos os sub-objetivos foram alcançados através do desenvolvimento/utilização de métodos que, tendo por base um registo *GPS*, são capazes de extrair quais as atividades realizadas, quais os lugares visitados e quais as parcelas marcadas, tendo por base um padrão a identificar no registo. A *interface* permite observar os resultados obtidos, possibilitando ao técnico consultar o relatório da sua atividade diária. A *interface* permite também a identificação/correção dos locais que não consigam ser identificados ou que possam ser incorretamente identificados. Toda a *interface*, implementada numa aplicação *web*, permite a interação do utilizador com todas estas funcionalidades.

1.5 Organização do Documento

O presente documento está divido em 7 capítulos:

Capítulo 1 Neste capítulo é apresentada qual a motivação, quais os objetivos, contribuições e o problema que se pretende resolver com esta dissertação.

Capítulo 2 Neste capítulo é descrita toda a literatura que foi revista associada a trabalhos efetuados por terceiros, assim como, toda a tecnologia utilizada para desenvolver a solução. É apresentada inicialmente uma secção relacionada com a captação de dados *GPS*, seguida de uma secção relacionada com a apresentação de um artigo sobre correções de coordenadas a registos *GPS*. Posteriormente, é introduzido o estado da arte relacionado com a deteção de lugares e identificação de atividades. Após a revisão da literatura é apresentada uma secção com as tecnologias que foram usadas ao longo do desenvolvimento da dissertação.

Capítulo 3 Ao longo deste capítulo é descrita a abordagem tomada para realizar a solução associada a esta dissertação.

Capítulo 4 Neste capítulo são descritos os aspetos de modelação relacionados com a descrição do sistema e base de dados que o suporta.

Capítulo 5 É descrita detalhadamente ao longo deste capítulo toda a implementação realizada ao longo do desenvolvimento da dissertação. Neste capítulo detalha-se toda a algoritmia, metodologia e funcionalidades realizadas para atingir a solução obtida com o desenvolvimento desta dissertação.

Capítulo 6 Neste capítulo é exposto e discutido todo o processo de avaliação da solução criada. São também apresentadas algumas conclusões associadas aos resultados obtidos.

Capítulo 7 Neste capítulo é apresentada uma apreciação geral do trabalho que foi realizado no âmbito desta dissertação, assim como algumas sugestões para alguns caminhos a seguir em relação a trabalho futuro.

2

Trabalho relacionado

Este capítulo é dedicado à descrição do trabalho relacionado, realizado por terceiros, correspondente às diferentes áreas que irão ser abordadas ao longo da dissertação. O estado da arte presente neste capítulo está dividido em quatro secções: a primeira secção (2.1), descreve os vários termos associados ao sistema *GPS*, a secção (2.2) descreve o trabalho já efetuado na área das correções feitas a dados obtidos por *GPS*. A secção (2.3) do capítulo, descreve as várias abordagens já realizadas na área da extração de lugares importantes, tendo por base dados provenientes de registos *GPS*. Após a análise aos mecanismos de extração de locais, a secção (2.4) resume a bibliografia relacionada com trabalho já efetuado na área tratamento de dados *GPS* para obter as atividades realizadas pelo utilizador. Por último, serão apresentadas algumas conclusões gerais do capítulo e as tecnologias que foram selecionadas para o desenvolvimento da solução.

2.1 Captação de dados - *GPS*

O sistema de posicionamento global (*GPS* - geo-posicionamento por satélite), é um sistema de navegação por satélite que permite obter uma localização, bem como o tempo (horas), em qualquer lugar do planeta desde que sejam visíveis, na área em que se está, um conjunto de quatro ou mais satélites. O sistema foi criado em 1973, tendo inicialmente um propósito militar, acabando por se tornar disponível também para o uso civil. Para um utilizador saber a sua posição, latitude e longitude, esta tem de ser determinada pelo recetor. A captura da posição através do *GPS* é feita em três passos, sendo eles: a receção do sinal dos satélites, a medição da distância e o cálculo da posição [3].

A receção do sinal *GPS*, pode demorar mais ou menos tempo, dependendo da situação de arranque do *GPS*. Ou seja, o tempo que o dispositivo recetor demora a identificar

os satélites depende do conhecimento *a priori* que este tem quando inicia o processo. Caso o recetor arranque a "frio", o que acontece normalmente na primeira utilização do dispositivo, o tempo de identificação será maior pois o recetor não tem qualquer noção da posição dos satélites. Considera-se um arranque "morno" se o dispositivo, quando arranca, já conhece as posições dos satélites. Por último, se o recetor guarda uma efeméride [1] válida, calcula as posições dos satélites de uma forma muito mais rápida e precisa, chamando-se a este processo arranque "quente".

O cálculo da distância é feito da seguinte forma: mede-se o tempo que o sinal demora a chegar do satélite até ao recetor, dividindo-se posteriormente pela velocidade de propagação do sinal. O cálculo das distâncias tem de ser feito usando pelo menos quatro satélites, três para calcular as posições em três dimensões e o quarto para sincronizar o tempo entre os satélites e o recetor. Por fim calcula-se a sua posição.

A localização pode ser calculada através de triangulação, ou seja, usando figuras geométricas (triângulos) para computar a posição. Uma triangulação pode ser feita através de *lateration* ou *angulation*. A primeira consiste na computação usando a distância de vários pontos conhecidos, neste caso, os satélites. Por outro lado, a *angulation*, baseia-se nos ângulos entre os vários pontos e o recetor.

Associado a uma posição *GPS*, estão vários atributos que determinam a qualidade de uma determinada coordenada obtida pelo dispositivo[16]:

Ruído: O ruído do sinal, pode afetar a precisão do cálculo da posição entre 1 a 10 metros e resulta, ou de uma interferência estática, ou do facto de haver mais comunicações numa determinada frequência. No entanto, o cálculo pode ser afetado até 3 vezes mais que o anterior por interferências de, por exemplo, edifícios, árvores ou montanhas.

***Position Dilution of Precision (PDOP)*:** A diluição da precisão da posição, indica a geometria dos satélites necessária à triangulação da posição do utilizador.

Neste sistema será utilizado um sistema de coordenadas geodésicas, latitude [15] e longitude [15]. A latitude geográfica de um ponto na superfície da Terra, equivale ao ângulo entre o plano equatorial e uma linha que passa por esse ponto e é normal à superfície de referência que aproxima a forma da Terra. A latitude mede-se para norte e para sul do Equador, entre -90º no polo sul e +90º no polo norte. A longitude, descreve a localização de um lugar medido em graus, de 0º a -180º para Oeste ou a 180º para leste, a partir do Meridiano de *Greenwich*. Por exemplo, a *FCT* tem uma latitude de 38.660140 e uma longitude de -9.203082.

[1] Os dados de efeméride contêm informações importantes sobre a situação de cada satélite (elementos de Kepler e parâmetros associados para compensar forças perturbadoras) como a data e a hora atuais.

2.2 Sistemas de correção de posicionamento

Em [13] o tema abordado prende-se com os erros que muitas vezes estão associados à obtenção da posição por parte dos dispositivos *GPS*. O problema abordado pelo artigo, está relacionado com as melhorias que têm de ser implementadas, para compensar a inexistência de mecanismos de correção de erros de posição nos dispositivos *GPS* dos utilizadores comuns, comparados com os que os dispositivos mais caros possuem. O exemplo no artigo tem por base o uso do GPS durante os passeios pedestres, e os erros finais que aparecem nos registos dos dispositivos resultantes das falhas de sinal originadas pelos edifícios altos ou pelo atravessamento de passagens estreitas.

Para a resolução deste problema, os autores propõem um sistema baseado em puro *software* dinâmico, desenvolvido em *Java*, aplicável a todos os dispositivos GPS e que tem por base um mecanismo de *Pipe and filtering*, ou seja, ao invés de a posição recebida ser enviada diretamente para os registos e ser apresentada ao utilizador, a posição recebida entra num mecanismo sequencial com vários filtros em que lhe é aplicado um determinado algoritmo cujo resultado passa para o filtro a seguir. No fim, a posição é ou não incluída nos registos. Esta sequência, é demonstrada na figura 2.1. Cada filtro está implementado num ficheiro próprio, que contém o algoritmo do filtro, fazendo com que a aplicação seja extensível, ou seja, é possível adicionar mais filtros sem ter de alterar o código original, bastando apenas introduzir o caminho para o ficheiro que contém o novo filtro.

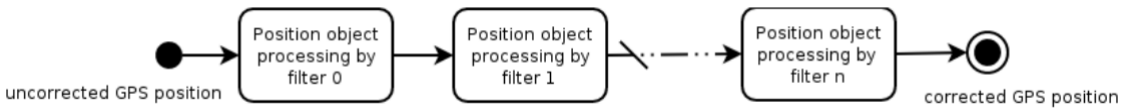


Figura 2.1: Sequência de filtros desenvolvida.[13]

Foram desenvolvidos e implementados quatro filtros. O primeiro filtro permite remover posições inválidas obtidas pelo *GPS*, o segundo consiste num filtro para especificar as exigências mínimas associadas à qualidade das várias características do sinal, o terceiro filtro é mais específico para passeios pedestres e, por fim, um filtro de remoção relacionado com a baixa precisão das mudanças de direção repentinas registadas pelo GPS, que por vezes são falsas.

O primeiro filtro tem por por nome *Invalid Position Filter*. Por vezes, os dispositivos *GPS* classificam uma determinada posição obtida como inválida, no entanto,nada impede que o próprio dispositivo a inclua no final nos registos e por fim a mostre ao utilizador. Como tal, o objetivo deste filtro é impedir que tal aconteça, descartando a posição, caso esta esteja assinalada como uma posição inválida.

O segundo filtro, *Mask Filter*, impõe que, caso se queira incluir uma determinada posição nos registos finais, os atributos associados a esta tenham de ter uma determinada

qualidade. Existem vários tipos de atributos associados a uma posição recebida pelo GPS, tal como descrito anteriormente. Como tal, um exemplo de atuação do filtro seria se, por exemplo, caso a *PDOP* fosse inferior a 6, o filtro atuaria pois este valor indica uma má classificação e o não respeito da máscara definida para esta qualidade. A posição seria considerada nula e descartada dos registos. No entanto, definir um valor estático para cada máscara seria uma má opção, dado que a qualidade do sinal é variável sendo, por vezes, apenas aceitável. Desta forma, o próprio filtro desenvolvido aprende com as várias posições que vai recebendo, adaptando o valor da máscara para um número aceitável.

A base da criação do terceiro filtro, *Map-Matching Filter*, tem origem num mecanismo que está incluído nos dispositivos GPS presentes nos automóveis. Este mecanismo consolida a posição recebida, por forma a que se esta é um ponto perto de uma estrada, e tendo em conta que os carros foram feitos para andar em estradas, automaticamente coloca o ponto na estrada mais perto. Como tal, o objetivo deste filtro é trazer este mecanismo para os dispositivos que as pessoas que praticam passeios pedestres possuem. Foi assumido, pelos autores, que uma pessoa que realiza este tipo de passeios não está sempre a entrar e a sair de edifícios ou lagos. Como tal, o filtro desenvolvido implementa um mecanismo que corrige a posição para que sempre que uma posição seja determinada como sendo interior a um lago ou um edifício, esta seja colocada num ponto perto da entrada do mesmo, mas no exterior deste. Esta situação está descrita na figura 2.2.

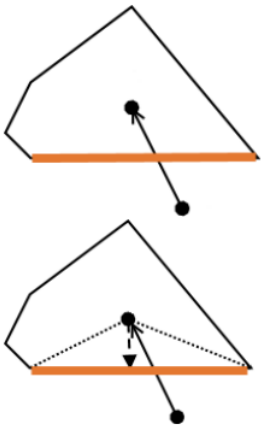


Figura 2.2: Correção da intersecção com edifícios.[13]

Por último, o filtro *Statistical Correction Filter*, permite a correção da posição *GPS* sempre que é detetado que o ângulo entre a posição anterior e a posição que está a ser avaliada é superior a 45 graus. O filtro altera a posição, que está atualmente a ser avaliada, para um meio termo entre a posição anterior recebida e a atual. Apesar do filtro conseguir compensar este tipo de mudanças, é sempre difícil saber se se está a agir bem ou não pois o utilizador pode realmente ter mudado de direção.

Os autores realizaram alguns testes sobre os resultados da aplicação dos filtros. Depois de avaliarem o sistema desenvolvido, após a integração deste num registo GPS de

um caminhante, os dados registaram uma correção no erro obtido de 12 quilómetros para 10 quilómetros. Nesta experiência, o primeiro filtro não foi aplicado pois não haviam posições inválidas. O menor erro foi obtido quando a ordem melhor de filtragem foi: *Invalid Position Filter* → *Mask Filter* → *Statistical Correction Filter* → *Map-Matching Filter*.

2.2.1 Conclusões

Sumarizando a secção, o importante a enfatizar é o contributo positivo que é trazido pelo sistema com a introdução dos filtros. Os filtros apresentados, permitem uma melhoria na qualidade dos registos obtidos no final de uma captação feita por *GPS*. No entanto, não foi necessária a implementação de nenhum destes filtros no sistema desenvolvido pois foram criadas outras formas de lidar com o problema associado ao erro e ao ruído.

2.3 Deteção de locais

Em [10], [19], [21], os autores decidiram fazer um aproveitamento dos dados espácio-temporais gerados pelas trajetória das pessoas ao longo dos dias para encontrar semelhanças entre utilizadores, tendo assim por objetivo encontrar similaridades entre os mesmos, baseando-se na sequência dos locais visitados por eles.

Os registos GPS iniciais inseridos na aplicação, obtidos pelos dispositivos GPS, vêm sobre a forma de uma sequência de pontos GPS. Um ponto recolhido pelo GPS é constituído por um conjunto de propriedades sendo elas a sua latitude, a sua longitude e uma data, que representa o momento em que foi recolhido determinado ponto. Assim, uma trajetória GPS é dada por uma sequência desses mesmos pontos ordenados temporalmente. Uma representação dos conceitos apresentados é a da figura 2.3, em que é apresentada uma sequência de pontos GPS (p1,p2...pn) e elaborada uma trajetória.

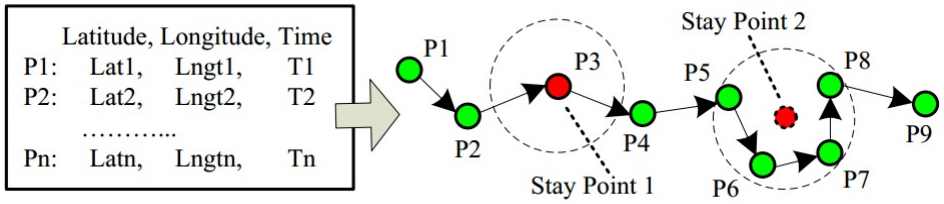


Figura 2.3: Pontos GPS e *stay points*. [19]

O último conceito introduzido no artigo, relacionado com esta matéria, está relacionado com a definição de Ponto de estadia (*Stay points*). Um ponto de estadia tem associado um significado semântico, ou seja, representa um lugar físico em que o utilizador esteve durante determinado tempo (um restaurante, a casa onde o utilizador vive...). Cada *stay point* extraído contém a hora a que chegou e a hora a que saiu. São apresentados 2 tipos de pontos de estadia:

Tipo 1: Representa um tipo de ponto de estadia, em que o utilizador está estacionário durante um determinado período num determinado ponto, o que faz com que este seja classificado como Ponto de estadia (Exemplo: o ponto P3).

Tipo 2: Desta vez, a classificação atribuída é baseada no facto de o utilizador circular dentro de uma determinada região, definida por um raio estabelecido, durante um determinado tempo. Assim, é calculado o centro de massa da região e atribuído um ponto de estadia neste (Exemplo: trajetória entre os pontos P5 e P8).

O algoritmo usado na parte experimental do artigo, para determinar os pontos de estadia, é apresentado como uma forma de pesquisar e obter, a partir dos registos iniciais, os pontos de estadia baseando-se em dois parâmetros: o tempo que é despendido numa determinada área ou ponto e o raio da distância que deve ser considerado para calcular os pontos de estadia do tipo 2. Por exemplo, na experiência em [19] a distancia escolhida foi de 200 metros e o tempo 30 minutos. A complexidade computacional do algoritmo é *O(n)*, em que *n* é o número de pontos GPS. No final, teremos uma estrutura que contém todos os pontos em que determinado utilizador permaneceu durante algum tempo.

Listing 2.1: Algoritmo para a deteção de *stay points*.[10]

```
Input: A GPS log P, a distance threshold distThreh and time span threshold
    timeThreh
Output: A set of stay points SP={S}

i=0, pointNum = |P|; //the number of GPS points
 while i < pointNum do,
   j:=i+1; Token:=0;
   while j < pointNum do,
    dist:=Distance(p_i, p_j); //calculate the distance between points
    if dist > distThreh then
      ΔT:=pj.T-pi.T;  //calculate the time span between two points
      if Δ T>timeThreh then
        S.coord:=ComputMeanCoord({p_k | i<=k<=j})
        S.arvT:= p_i.T; S.levT=p_j.T ;
        SP.insert(S);
        i:=j; Token:=1;
        break;
    j:=j+1;
    if Token!=1 then i:=i+1;
return SP.
```

Após a extracção dos *Stay Points*, será necessário agregar os mesmos em vários conjuntos, de forma a obter os locais finais. A este procedimento chama-se *clustering* ou agrupamento. Existem dois tipos de agrupamento (*clustering*): por Partição e por Densidade.

2.3.1 *Clustering:* por partição

Em [2], o objetivo dos autores foi desenvolver um sistema capaz de agregar vários pontos GPS obtidos através da rotina de um utilizador de forma a conseguir obter localizações físicas reais. Assim, o artigo é relevante para este trabalho por duas razões. Em primeiro lugar, por abordar uma forma diferente de detetar locais e em segundo lugar, pelo método de *clustering* utilizado para agregar os pontos GPS em localizações.

A forma de deteção de um ponto importante apresentada neste artigo, está relacionada com a identificação do momento da perda do sinal GPS. Sempre que a diferença, entre a obtenção de dois pontos consecutivos, seja superior a um tempo t, considera-se que o utilizador esteve num "lugar". Após serem determinados os pontos que se incluem no padrão descrito, procede-se a uma operação de *clustering* para determinar o local real que se pretendia. O tempo t escolhido pelos autores foi de 10 minutos após verificarem que era o tempo médio que um utilizador permanecia dentro de um edifício. No entanto, fica ao critério de quem implementar o algoritmo.

Para a tarefa de *clustering*, o algoritmo utilizado é o algoritmo por partição *K-means* [7], que é um algoritmo iterativo de *clustering* bastante eficiente que objetiva particionar n observações dentre k *clusters* em que cada observação pertence ao *cluster* mais próximo da média. Ao longo do resto do artigo é utilizada a *framework* desenvolvida e apresentados os resultados.

Apesar da eficiência do algoritmo, demonstrada no artigo, existem alguns problemas associados ao uso deste algoritmo. Em primeiro lugar, o número de *clusters* (localizações) tem de ser dado como *input* ao algoritmo, o que significa que teríamos de saber o número de lugares *a priori* a classificar antes de sequer sabermos quantos iríamos encontrar. Em segundo lugar, dado que neste algoritmo todos os pontos GPS têm de pertencer a algum *cluster*, este algoritmo é muito propício a ruído o que nos pode levar a más classificações a quando da determinação das localizações, ou seja, um lugar pouco importante e longe dos outros pode causar um grande afastamento da localização que se pretendia.

2.3.2 *Clustering:* por densidade

A potencialidade atribuída aos algoritmos baseados em densidade faz com que estes sejam bons candidatos quando existe a necessidade de executar o *clustering* de informação. Assim, em [23], é introduzido um novo algoritmo capaz de resolver as limitações dos problemas associados ao *clustering* para obtenção de lugares importantes, sendo este um dos motivos principais da citação do mesmo. Para além das limitações já apresentadas em cima, associadas aos algoritmos por partição, o uso de algoritmos baseados em densidade permite descobrir *clusters* de formas irregulares, ao contrário dos algoritmos por partição em que estamos limitados às formas esféricas. É possível verificar o conceito apresentado na figura (2.4), em que cada ponto corresponde a um *stay point* e o contorno à volta de alguns corresponde à unificação de todos os *stay points* envolvidos num único lugar real, sendo escolhido um ponto do grupo para indicar as coordenadas do local.

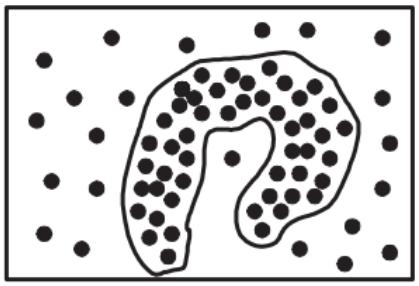


Figura 2.4: Agregação de pontos num único grupo através de *clustering* por densidade [23].

Outra das razões, apresentada pelos autores, para a utilização de *clustering* por densidade, é que estes algoritmos, como são menos sensíveis a ruído, diminuem a probabilidade da inclusão nos resultados de situações que não interessam, como pontos muito afastados ou apenas pontos esporádicos. Por último, embora estes algoritmos necessitem de parâmetros de densidade, como a distância que se deve ter em conta na procura e o número de pontos mínimo em cada *cluster*, a variação destes, de aplicação para aplicação, provavelmente será nula. Ao contrário do que acontece nos algoritmos por partição, aqui o resultado é sempre o mesmo.

Um dos algoritmos mencionados no artigo é o *DBSCAN* [5], um dos algoritmos mais importantes neste tipo de *clustering*. No entanto, segundo os autores, não é uma boa opção devido ao desempenho e sensibilidade a ele associados.

A parte mais relevante do artigo é a introdução e descrição integral do algoritmo *DJ-Cluster*. O *DJ-Cluster* [22] é um novo tipo de algoritmo determinista baseado em densidade que, depois de terminar retorna, para todos os pontos incluídos na estrutura de pontos passada como argumento, uma atribuição de cada um desses pontos a um *cluster*. Ou seja, no final, o que é obtido são, não as localizações que tínhamos inicialmente mas sim a sua representação real, ou seja, os lugares reais concretos. O comportamento do algoritmo é o seguinte: para cada ponto recebido, este verifica a quantidade de pontos dentro de uma vizinhança a uma determinada distância do ponto que se está a analisar, representada por *Eps*, verificando também se o número total de pontos encontrado é superior ou igual ao número mínimo dado como parâmetro inicial, neste caso *MinPts*. Caso o número de pontos seja superior ao dado no parâmetro, os pontos passam todos a fazer parte de um novo grupo, senão o ponto que estava a ser testado é marcado como ruído. Caso os pontos que foram descobertos na vizinhança já estivessem incluídos em algum *cluster*, esse é unido ao descoberto na iteração atual. No final, a localização real de um determinado *cluster* é o ponto que, dentro desse mesmo *cluster*, obteve um maior número de vizinhos a uma determinada distância (*Eps*).

Referem também os autores que, tanto *Eps* como *MinPts*, devem ser atribuídos consoante a necessidade do problema. Neste caso, os valores são 30 metros para a distância, sendo o segundo parâmetro deixado ao critério de quem está a usar o algoritmo. Um

valor de *Eps* muito alto pode resultar na junção de vários lugares num único, pois o raio da área de procura é maior, o que faz com que vários pontos passem a pertencer a um mesmo *cluster*. Para o parâmetro *MinPts*, o valor decidido tem por base o mesmo raciocínio, ou seja, caso *MinPts* seja um valor muito baixo pode resultar num número maior de pontos num *cluster* e o inverso causando a situação contrária.

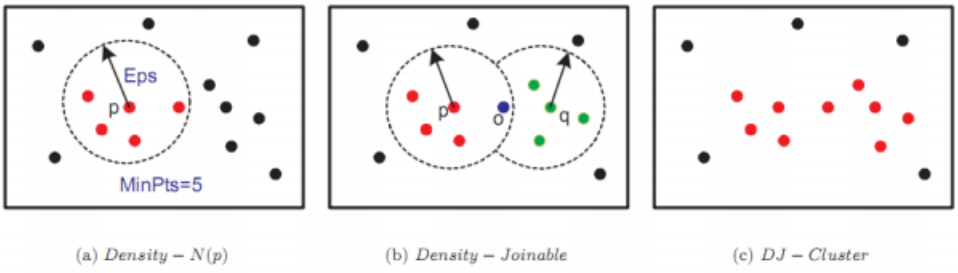


Figura 2.5: Agrupamento com base em densidade. (a) ilustra uma vizinhança N do ponto p; (b) ilustra que N(p) e N(q) são acopláveis por densidade; (c) ilustra o grupo final em cor vermelha. [22]

As definições do algoritmo são as apresentadas em baixo.

Definição: A vizinhança N baseada em densidade de um ponto p denotada por $N(_p)$, é definida por 2.1 e 2.2

$$N(p) = \{q \in S | dist(p,q) \leq Eps\} \tag{2.1}$$

$$\|N(p)\| \geq MinPts \tag{2.2}$$

em que *S* é o conjunto de todos os pontos, *q* é um ponto representativo da amostra, *Eps* é o raio de um círculo à volta de *p* que define a densidade e *MinPts* é o número mínimo de pontos necessários no círculo em causa.

Definição: (junção por densidade) *N(p)* é densamente agrupável com *N(q)* se, para um determinado valor de *Eps* e um valor de *MinPts*, ambos contiverem um ponto *o* em comum, tal como é apresentado na figura 2.5. O algoritmo é apresentado em 2.2.

Listing 2.2: Algoritmo *DJ-Cluster*.[22]

```
Perform all temporal data pre-processing.
Select an unprocessed point p from sample S.
if p is null then
Return
end if
Compute the density-based neighborhood N (p) of a
point p wrt Eps and MinP ts.
if N (p) is null then
Label p as noise.
else if N (p) is density-joinable to at least one existing
cluster then
```

```
12 Merge N (p) and all the density-joinable clusters.
13 else
14 Create a new cluster C based on N (p).
15 end if
16 Return to step 2.
```

Principais propriedades do algoritmo 2.2:

- Todos os pontos têm de pertencer a algum *cluster*, senão são marcados como ruído;
- Existe pelo menos um ponto em cada *cluster*;
- O algoritmo particiona os dados de entrada em grupos não-hierárquicos;
- O *clusters* não se sobrepõem.

Apesar da eficiência do Algoritmo *DJ-Cluster*, durante a atribuição dos vários pontos aos vários *clusters*, o algoritmo, na avaliação de um ponto, só tem em conta a frequência ($MinPts$) dada uma determinada distância de procura (Eps), não sendo tida em conta a duração despendida no mesmo. Assim, em [4] é proposta uma alteração ao algoritmo por forma a que este passe a ter em conta também a duração despendida numa determinada localização. O autor propõe assim que a condição de inclusão passe a ser a descrita em (2.3), mantendo todo o resto do algoritmo igual.

$$N(p) \geq MinPts \vee N(p).duration \geq MinDrt \tag{2.3}$$

2.3.3 Conclusões

Em suma, nesta secção foram apresentados os mecanismos que permitem, a partir de um registo inicial de dados *GPS*, extrair locais reais visitados pelo utilizador. É descrita inicialmente uma forma de fazer um pré-processamento dos dados utilizando a extração de pontos de estadia (*Staypoints*). O resultado da aplicação desta, consiste numa sequência de pontos já tratados que, por vezes, tal como foi apresentado na figura (2.3), representam regiões em que estão incluídos vários pontos.

Posteriormente, são analisados os vários tipos de *clustering* existentes. Perante a análise feita, tudo indica que o uso do algoritmo baseado em densidade *DJ-Cluster*, consequente do facto de ser menos sensível a ruído, permite definir locais com forma arbitrária e não necessita do número de locais reais como parâmetro inicial, ou seja, o número de *clusters*, como uma informação *a priori*. Tal como é descrito, no final da fase de *clustering*, são já conhecidas as coordenadas dos locais reais finais.

Por fim, é explorada uma abordagem, apresentada numa dissertação, que sugere uma alteração nos parâmetros do algoritmo *DJ-Cluster*, por forma a que este não tenha em conta, em demasia, a densidade, ou seja, a frequência. Por exemplo, se um utilizador ficar bastante tempo num local e nunca mais lá voltar, o algoritmo passará a ter em conta este mesmo local.

2.4 Deteção de Atividades

Como foi dito anteriormente na introdução deste documento, os *smartphones* tornaram-se numa grande fonte de informação contextual devido aos vários sensores que têm embutidos. Em [9], os autores propõem um sistema em que a partir da informação contextual produzida pelos *smartphones*, é possível extrair informação sobre as atividades que o utilizador efetuou, inferindo as mesmas usando *hierarchical Bayesian networks*.

O sistema desenvolvido pelos autores é constituído por quatro partes: a primeira e a segunda parte estão relacionadas com a captação dos dados e o pré-tratamento aplicado aos mesmos. Posteriormente, a informação é submetida a uma rede *bayesiana* e os resultados são mostrados ao utilizador através do último componente, a *interface* do sistema. A relevância na citação deste artigo, está relacionada com o mecanismo de inferência das atividades usado pelos autores, que tem por base o uso de redes de probabilidades condicionais.

Segundo os autores, o primeiro passo para poder inferir as atividades, é decidir qual a informação contextual relevante. A figura 2.6, apresenta os vários tipos de contexto que poderão identificar uma atividade sob a forma de um modelo. O modelo relaciona a inferência de uma determinada atividade, com o conjunto de contextos (espaciais, temporais, sociais ..) que a caracteriza. Ou seja, mostra a dependência entre o contexto e as observações, como por exemplo a localização, dia da semana ou as horas, e como tudo isto se relaciona com a atividade inferida.

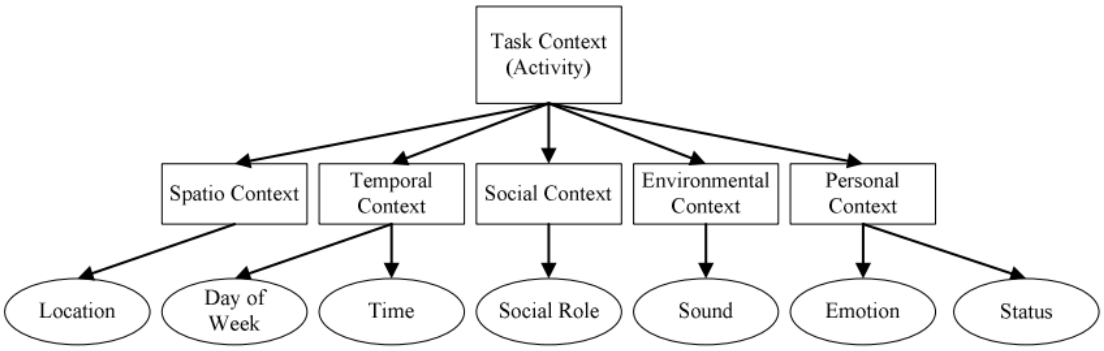


Figura 2.6: Modelo contextual para o reconhecimento de uma atividade [9]

Porém, devido ao facto de este ser um processo de predição em que as conclusões a retirar dependem de valores condicionais, os autores decidiram usar uma rede de *bayes*, para representar as dependências probabilísticas associadas às variáveis, que correspondem neste caso às observações do *GPS*. Uma rede de *bayes*, é modelada através de um grafo direcionado e acíclico, no qual os nós representam variáveis (discretas ou contínuas), e os arcos representam a dependência condicional ou informativa entre as variáveis, ou seja, os nós. Para representar a relação condicional, a rede baseia-se no teorema

de *bayes*(2.4) e na independência condicional entre as variáveis. A rede especifica, implicitamente, a distribuição de probabilidade conjunta de todas as variáveis, a partir da distribuição condicional de probabilidade de cada variável, dados os seus pais [14].

$$P(X|Y) = P(X)\frac{P(Y|X)}{P(Y)} \tag{2.4}$$

Aplicando a definição ao problema apresentado no artigo, os nós na rede representam, por um lado, o conjunto de observações como o local ou o dia-da-semana, e por outro, as atividades inferidas como sair ou jantar. As observações são denotadas por O={o_1, o_2, o_3, o_n}, sendo que as atividades são denotadas por A={a_1, a_2, a_3, a_m}. Os nós, em conjunto com a estrutura de arcos que lhes está associada, definem a probabilidade condicional P($A \mid O$). Ou seja, a probabilidade de ser inferida a atividade A, dadas as observações O (2.5).

$$P(A|O) \rightarrow P(a_i|o_1, o_2, o_3, o_n) \tag{2.5}$$

O modelo da estrutura da rede de *bayes* é apresentado na figura 2.7.

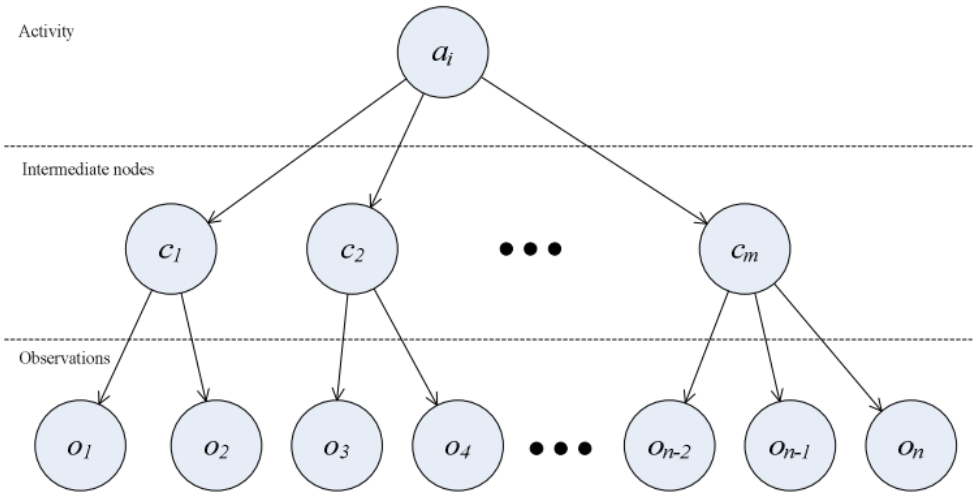


Figura 2.7: Estrutura da rede de *bayes* para reconhecimento das atividades [9]

O artigo termina com a apresentação de resultados. As 23 atividades que a rede consegue inferir, foram definidas com base num inquérito estatístico da *GSS (General Social Survey on Time Use, 1998)*. Segundo os autores, a inferência de atividades, usando os modelos de redes *bayesianas*, foi um sucesso na maior parte das inferências que eram esperadas pelo sistema desenvolvido. Conseguiu-se classificar atividades como por exemplo:

Dormir à noite: Acontece especificamente em casa (contexto local), e maioritariamente à noite (contexto temporal).

No entanto, para atividades como por exemplo relaxar ou ler livros, o algoritmo obteve uma percentagem de classificação um pouco aquém do esperado. Em 23 atividades, o algoritmo conseguiu ter uma margem de precisão acima dos 50% em 15 das 23 atividades.

Um outro método de classificação foi utilizado em [11]. No artigo, os autores apresentam uma outra forma de inferir atividades, estando a relevância do mesmo relacionada com a apresentação de uma *framework* baseada em *Relational Markov Networks* (RMNs) [17], que tem como objetivo extrair informação sobre atividades do dia-a-dia de um utilizador, tais como, jantar, ir às compras ou jantar fora.

Uma *RMN* é uma extensão dos *Conditional Random Fields* [8], que são modelos baseados em grafos unidirecionais cujo objetivo é classificar informação, sendo estes de grande utilidade na classificação de atividades [12]. Uma *RMN* é constituída por quatro partes fundamentais: um *Schema* (ε), uma instanciação (I), um conjunto de modelos de cliques relacionais (*relational clique templates* C) e por fim os seus respetivos potenciais (Φ). A melhor forma de entender os três primeiros conceitos apresentados, é fazendo uma comparação com um modelo relacional de base de dados. Um *Schema*, corresponde à especificação das várias tabelas de uma base de dados bem como dos vários atributos. Uma instanciação, corresponde à informação que é colocada numa base de dados, e por último, os *relational clique templates*, correspondem às consultas *SQL*. Um *template* de um clique relacional pode resultar em vários cliques, ou seja, um clique é um sub-conjunto de um grafo denotado por um conjunto de nós, neste caso os vários resultados de uma consulta aplicada à *RMN* originam um conjunto de cliques. Por exemplo, no modelo apresentado, é possível construir um *clique template* sobre todas as atividades que foram marcadas com a etiqueta "AtHome", o que pode resultar num conjunto de respostas que são chamadas de cliques.

A cada clique resultante de uma determinada consulta, é associado um valor de potencial, Φ. Um valor de potencial, é simplesmente uma tabela de valores para cada nó que pertence à clique que se está a analisar, e define a "compatibilidade" entre os valores dos atributos do clique. Este valor é calculado usando a formula (2.6).

$$\Phi_c(V_c) = exp\{w_c^T . f_c(v_c)\} \tag{2.6}$$

Uma das componentes da formula são os pesos w, e estão associados a cada atributo de uma variável v do clique. O peso determina a importância que a característica $f_c(v_c)$ tem no resultado final do potencial. Assim, para uma instanciação específica I, uma *RMN* em [8] define um distribuição condicional de P(Y | X), em que Y são as atividades e X os vários atributos que pertencem às várias classes do *Schema*. A probabilidade é calculada pela multiplicação dos potenciais de cada clique resultante da consulta efetuada, em conjunto com uma função de normalização(2.7).

$$P(Y|X) = \frac{1}{Z(x)} \prod_{c \in C} \prod_{v_c \in} \Phi_c(v_c) \tag{2.7}$$

O *Schema* definido pelos autores é composto por três classes: Atividades, lugares e transições.

Activity: A classe de atividade é a principal do domínio. O atributo *label* ou etiqueta, é a sua única variável oculta, ou seja, tem de ser estimada a partir das outras. As etiquetas possíveis, ou seja, as várias atividades que podem ser identificadas são: "AtHome", "AtWork", "Shopping", "DiningOut", "Visiting", "Others". O atributo *Id* é o atributo usado como chave primária. A classe contém também alguma informação temporal relacionada com a atividade, como o *TimeOfDay*, *Day-OfWeek*, e *Duration*, ou seja, o período do dia em que aconteceu, o dia da semana e a duração da mesma. Finalmente, *Place* é a chave externa para a classe dos lugares.

Place: A classe *Place* ou lugar, inclui dois atributos *booleanos*: *NearRestaurante* e *NearStore* (atributos usados no exemplo do artigo), que indicam se existem ou não restaurantes ou lojas perto do lugar.

Transition: A classe *Transition* ou transição, contém toda a sucessão temporal. Possui os atributos *From* e *To*.

Com base no *Schema* apresentado, os autores definiram os seguintes *relational clique templates*:

Padrões temporais: Diferentes atividades têm informações temporais diferentes como por exemplo a parte do dia em que ocorrem.

Evidências geográficas: Informações sobre o tipo de restaurante.

Relações de transição: As transições entre as atividades pode ser importante. Por exemplo, ir de casa para o trabalho é uma transição comum. Um exemplo de *SQL* poderia ser a consulta em 2.3.

Condições espaciais: Atividades que ocorrem no mesmo lugar são normalmente similares.

Listing 2.3: Consulta *SQL* exemplo de uma *clique template* [11]

```
1 SELECT a1.Label, a2.Label
2 FROM Activity a1, Activity a2, Transition t
3 WHERE t.From=a1.Id AND t.To=a2.Id
```

Concluindo esta parte do artigo, a probabilidade P(Y | X) apresentada na listagem 2.3, relaciona a probabilidade de acontecer uma determinada atividade Y, das que foram definidas no domínio da classe *Schema*, e as observações X.

A inferência de atividades nas *RMN's* tem por base o uso de *Markov chain Monte Carlo*(*MCMC*) [6]. Uma *MCMC* é um caso particular de processo estocástico por simulação, em que os estados anteriores são irrelevantes para a predição dos estados seguintes,

desde que o estado atual seja conhecido. No entanto, no artigo é referido que só por si a modelação do problema numa *MCMC* não é suficiente, optando por métodos de inferência mais complexos.

Como foi dito anteriormente, os pesos atribuídos a cada um dos atributos das várias classes do *Schema*, determinam o valor do potencial que é atribuído às várias cliques resultantes do mesmo *clique template*. Estes pesos precisam de ser estimados. Assim, são abordados vários métodos de aprendizagem baseados em aprendizagem supervisionada.

Para avaliar o desempenho do sistema, foi feita uma experiência com base em 2 conjuntos de dados: o primeiro de uma única pessoa, contendo aproximadamente 400 visitas a 50 locais diferentes num período de 4 meses e, para o segundo conjunto de dados, um conjunto de 25 a 35 visitas a, aproximadamente, 15 lugares diferentes mas desta vez com um conjunto composto por 5 pessoas. Depois de submetidos os dados à *framework* e retirados os lugares importantes, foram usados métodos de validação como o *leave-one-subject-out cross-validation* e quando treinada a informação conseguiram-se resultados na ordem de margem de erro de 7%.

2.4.1 Conclusões

Em suma, nesta secção foram apresentados os mecanismos que permitem a partir de um registo inicial de dados *GPS*, inferir quais as atividades realizadas pelo utilizador. Foram apresentadas duas formas de fazer a inferência: ou por meio de uma rede de *bayes* ou por meio de uma *RMN*.

Comparando os dois métodos, ambos obtiveram bons resultados. É indicado no segundo artigo analisado, que as *RMN* são muito mais fiáveis quando existe a necessidade de fazer este tipo de inferência, no entanto, a complexidade do sistema é muito superior em relação à rede de *bayes*.

2.5 Tecnologias

Ao longo desta secção serão apresentadas as várias tecnologias que serviram de base ao desenvolvimento de toda a dissertação.

2.5.1 PostgreSQL

O PostgreSQL [2], atualmente na versão 9.2.2, é um *(SGBD) open source* que, apesar da sua origem académica, é capaz de lidar com alguns mecanismos complexos presentes nos *SGBD* de maior porte e capaz de lidar com bases de dados de grandes dimensões. A aposta neste *SGBD* prende-se também com o facto do *PostgreSQL* ter uma extensão que permite dar um bom suporte a dados georreferenciados, o *PostGIS* [3].

[2] http://www.postgresql.org
[3] http://postgis.refractions.net/

Através do uso da extensão para dados geográficos, é possível mapear e guardar toda a informação relacionada com tipos geométricos numa base de dados que tem de raiz o suporte para os mesmos. Com o *PostGIS*, a base de dados passa a ter suporte para representar polígonos, multi polígonos, pontos, linhas, multi pontos, ou conjuntos geométricos que permitem aglomerar vários tipos geométricos, as *geometrycollections*. Para além da representação dos vários tipos geométricos, esta extensão disponibiliza um conjunto de funções que facilitam operações como: determinar a área ou o perímetro de um determinado polígono. De modo a permitir um acesso mais rápido à informação, o *PostGIS* implementa os índices do tipo *R-tree*, que são mecanismos que permitem indexar dados georreferenciados, aumentado a rapidez das consultas à base de dados.

2.5.2 Android

O *Android* [4] é um sistema operativo baseado em *Linux* desenvolvido pela *Google* [5]. A última atualização coloca-o na versão 4.2.1, *Jelly Bean*. É uma tecnologia *open source*, com bastante documentação, comunidade e está presente em vários *smartphones*. Toda a parte de aquisição de dados será feita usando este *SO*. O sistema operativo possui uma *API* que permite a interação com o sensor GPS presente no equipamento, bem como de outros sensores, como o acelerómetro, casos os possua.

2.5.3 Java

A linguagem *Java* [6], é uma linguagem de programação orientada aos objetos desenvolvida na década de 90 pela empresa *Sun Microsystems*. Atualmente, pertence à *Oracle* [7] e é uma das linguagens mais desenvolvidas e utilizadas atualmente em todo o mundo. Existe um conjunto de vantagens em usar esta linguagem para desenvolver os algoritmos necessários durante a dissertação. Em primeiro lugar, dado o facto de ter sido a linguagem mais utilizada em todo o curso, garante um mais à vontade na sua utilização. Por outro lado, o *Java*, devido ao facto de ser compilado para a sua própria máquina virtual, preza pela portabilidade, ou seja, independentemente da plataforma em que está a ser executado, seja Windows, Linux ou qualquer outro sistema operativo que suporte a máquina virtual do *Java*, garantidamente que que é possível executar o programa. Outra grande vantagem da linguagem é o mecanismo de gestão automática, o *garbage collector*, que permite a alocação e libertação de memória automaticamente.

[4] www.android.com
[5] http://www.google.com/about/company/
[6] http://www.oracle.com/java
[7] http://www.oracle.com

2.5.4 Java EE

A plataforma *Java* edição empresarial[8] é uma plataforma de programação para servidores que usa a linguagem *Java*, com contribuições de especialistas do setor, organizações comerciais e *open source Java*, grupos de usuários, e inúmeras pessoas. A plataforma oferece uma *API* e um ambiente para o desenvolvimento e execução de softwares empresariais, permitindo assim criar, entre muitas outras funcionalidades, serviços *web*.

2.5.4.1 Hibernate

O *Hibernate*[9], é uma biblioteca *ORM (Object-relational mapping)* para a linguagem *Java* que oferece um conjunto de métodos que possibilitam o acesso a uma base de dados, facilitando o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de ficheiros (*XML*) ou anotações *Java*. A biblioteca oferece um dialeto *SQL* próprio que permite executar consultas. Usando esta biblioteca e a sua extensão espacial, o *Hibernate Spatial*[10], obtém-se uma extensão às funcionalidades originais e passa a ser possível lidar com dados espaciais tais como, polignos, pontos ou linhas.

2.5.4.2 Jersey

O *Jersey*[11], é mais uma das bibliotecas disponibilizadas na versão empresarial *Java*, que permite a criação de uma interface *REST*, possibilitando assim criar serviços *web* que permitem aceder à aplicação desenvolvida.

2.5.5 Tecnologias de Informação Geográfica

2.5.5.1 GPX

GPX [12], ou *GPS eXchange Format*, é um *XML schema* que descreve um formato de um ficheiro *XML* que permite a troca de dados *GPS*. O formato *GPX* é o formato mais usado atualmente em serviços de cartografia ou mesmo *geocaching*.

Através deste tipo de formato, é possível descrever toda a informação sobre pontos, sequências de pontos ou rotas, permitindo assim que estas sejam mostradas num mapa. Toda a informação *GPS* que foi usada ao longo do projeto, encontrava-se inicialmente num ficheiro *GPX*.

De todos os elementos presentes num ficheiro *GPX*, existem 3 a salientar pela sua importância nesta dissertação. O elemento *track*, o elemento *waypoint(wpt)* e o elemento *trackpoint*. O elemento *track* é um dos elementos gerais de um *GPX* e representa uma

[8] http://www.oracle.com/technetwork/java/javaee
[9] http://www.hibernate.org/
[10] http://www.hibernatespatial.org/
[11] https://jersey.java.net
[12] http://www.topografix.com/gpx.asp

captação *GPS*, ou seja, uma *track* corresponde a um troço e é definida por uma sequência de pontos. Aos pontos que definem uma *track* é dado o nome de *trackpoints*. Dado que cada *trackpoint* guarda, para além da sua coordenada, qual a hora a que foi captado, ordenando todos os *trackpoints* captados sequencialmente pela sua hora de captura, obtemos a captação feita. Por outro lado, os *waypoints* representam pontos de interesse e são pontos que permitem salientar determinadas ocasiões ou lugares importantes numa captação. Cada *waypoint* possui também uma coordenada e uma hora de captura.

2.5.5.2 OpenStreetMap

O *OpenStreetMap* [13] é um projeto da *OpenStreetMap Foundation* criado em 2004 cujo slogan é "*The Free Wiki World Map*". Assim, a ideia principal deste projeto é criar um mapa do mundo todo com informação geográfica proveniente dos utilizadores, ou seja, todo o mapeamento e informação existente para consulta no *OSM* foi fornecida pelos utilizadores. Para além de ser possível criar um mapa através das informações do *OSM*, através das suas *Tiles*, este possui também um serviço que permite obter informação, o *Nominatim* [14]. O *Nominatim* é uma ferramenta que permite procurar informação sobre determinada localização a partir de nomes de cidades ou ruas, ou através de *Reverse Geocoding*, que consiste na procura de informação de uma dada localização dando simplesmente as suas coordenadas.

2.5.5.3 Google Maps

O *Google Maps* [15] é um serviço criado pela *Google*, cujo objetivo é fornecer informação cartográfica, tanto a nível de apresentação em mapa, como a nível de base de dados geográfica. A relevância em usar este serviço, está relacionada com a quantidade de informação que o *Google Maps* dispõe em termos de conhecimento de lugares. Tal como o *GeoNames* e o *Nominatim*, a *Google* possui também um serviço de *Geocoding* [16] que permite obter informação associada aos lugares, dadas as coordenadas. Este serviço fornece uma *API* que permite questionar a base de dados, no entanto tem algumas limitações a nível de pedidos por dia.

Um outro serviço oferecido pela *Google*, embora também limitado no número de acessos diários, é o *nearbysearch* [17]. O serviço permite pesquisar estabelecimentos de um determinado tipo próximos de uma determinada localização. Ou seja, indicando uma coordenada e o tipo de estabelecimento que se pretende, o serviço *nearbysearch* retorna quais os estabelecimentos mais próximos baseados na sua popularidade.

[13] http://www.openstreetmap.org
[14] http://wiki.openstreetmap.org/wiki/Pt-br:Nominatim
[15] https://developers.google.com/maps/
[16] https://developers.google.com/maps/documentation/geocoding/
[17] https://developers.google.com/places/documentation/search?hl=pt

2.5.5.4 MapQuest

O *MapQuest* [18] é um serviço grátis de mapas americano pertencente à *AOL*. A plataforma *MapQuest* permite a integração de mapas em qualquer ambiente seja ele mobile, web ou *desktop*, fornecendo informação sobre mapas de diferentes sítios bem como direções para lugares. É bastante usado para quando se pretende viajar pois disponibiliza informações sobre hotéis, restaurantes, etc...

A plataforma disponibiliza três tipos de licença. O primeiro tipo é mais direcionado a empresas, sendo que o segundo e o terceiro são mais para a comunidade em geral e relevantes para a dissertação. Por um lado, o *Community Edition (Licensed Data)* tem algumas limitações, embora ofereça em termos de conteúdo o mesmo que o *Enterprise*. Por outro lado, o terceiro tipo de licença, *Community Edition (Open Data)*, é baseado na informação disponibilizada no *OpenStreetMap*. A desvantagem em usar o *Community Edition (Licensed Data)*, está na limitação do número de acessos, caso estes sejam grandes. O *MapQuest* fornece um conjunto de *tiles*, baseados na informação do *OSM* mas com mais alguma informação obtida pelas fontes da própria plataforma. Para além desta ferramenta, a plataforma disponibiliza também uma *API javascript* que permite consultar o *Nominatim* e a própria base de dados do *MapQuest*, no caso da *Licensed Data*, obtendo a informação em formato *GIS* direto, podendo assim ser representada num mapa.

2.6 Conclusão

Ao longo deste capítulo foram descritas as várias técnicas assim como os vários trabalhos efetuados por terceiros na área da deteção de lugares e de atividades e por outro, as várias tecnologias que serviram de suporte ao desenvolvimento desta dissertação.

Para a deteção de lugares, após consideradas as várias técnicas, decidiu-se utilizar a técnica de deteção dos pontos de estadia e o agrupamento (*clustering*) por densidade. A escolha baseia-se não só nos bons resultados obtidos pelas técnicas, tanto nos artigos como na dissertação que é apresentada em [4], mas também na simplicidade necessária às suas implementações que se adaptam ao tempo disponível para a realização desta dissertação. Utilizando a técnica dos pontos de estadia é possível determinar os vários pontos importantes onde o utilizador despendeu algum tempo. Por outro lado, usando o agrupamento por densidade através do algoritmo *DJ-Cluster*, é possível agrupar os mesmos pontos de estadia e determinar os lugares onde o utilizador esteve. Usando o algoritmo de agrupamento por densidade descarta-se a necessidade de dar como parâmetro de entrada ao algoritmo o número de lugares, sendo assim possível obter no final a localização dos lugares que se esperam. Foi também utilizada a alteração proposta em [4] presente no final da secção 2.3.2, para possibilitar mais uma condição para a deteção dos lugares.

Devido à complexidade dos vários algoritmos apresentados para a identificação de atividades, bem como ao facto de as atividades que se pretendem identificar respeitarem

[18] http://www.mapquest.com

padrões próprios criados para esta dissertação, não foram usados quaisquer métodos dos apresentados. A deteção de atividades é feita através de algoritmos e metodologias que foram criados ao longo da dissertação e que permitem identificar os padrões concebidos para as atividades mencionadas na secção dos objetivos deste documento.

3

Abordagem

Ao longo deste capítulo será descrita a abordagem tomada para desenvolver uma solução capaz de cumprir os objetivos enumerados na secção 1.3. Na figura 3.1 apresenta-se um diagrama que representa o ciclo de desenvolvimento das várias fases da aplicação. Em primeiro lugar extraem-se os vários locais, depois são extraídas as várias atividades e por fim toda a informação obtida é mostrada numa interface desenvolvida para esta dissertação. Para além de permitir a visualização da informação é nesta que é possível corrigir a informação que tenha, por ventura, sido mal identificada. Toda a algoritmia e pormenores de cada fase serão descritos detalhadamente mais à frente no documento.

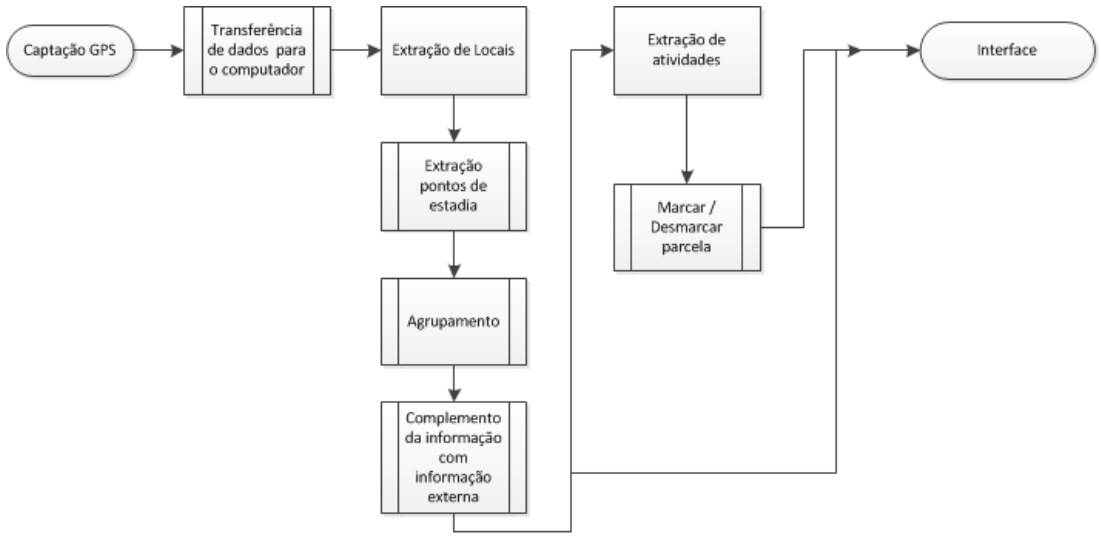


Figura 3.1: Diagrama geral da abordagem para a construção da solução

3.1 Aplicação Móvel

O ponto de partida para a extração das atividades realizadas pelo técnico, são as captações *GPS* obtidas pelo seu dispositivo móvel. A captação é realizada através de uma aplicação *Android* adaptada que permite registar todos os movimentos praticados pelo utilizador ao segundo, no período em que este mantém a aplicação a executar no seu *smartphone*. Fazendo uso do sensor *GPS*, o registo final consiste num conjunto de pontos ao qual estão associados, para cada um, uma coordenada, informação sobre a sua hora de captura e a precisão com que esta foi obtida (Figura 3.2). Como representado na figura, todo o processo tem início numa captação *GPS* cujos dados transferidos posteriormente para o *Desktop* no qual tem início a extração de informação. A linha a azul representa a união dos vários *trackpoints* obtidos pela captação.

Figura 3.2: Sequência inicial *GPS*

3.2 Deteção de lugares

Depois de carregados os dados da captação do utilizador do *smartphone* para o computador, é iniciado o processo de descoberta dos lugares visitados pelo utilizador durante o período da captação. Os lugares são extraídos usando as técnicas descritas em 2.3. O primeiro procedimento é restringir os milhares de pontos iniciais a um conjunto menor de pontos usando a técnica dos *Staypoints* ou pontos de estadia (Figura 3.3). Os pontos a amarelo na figura representam os vários pontos de estadia obtidos.

Figura 3.3: Pontos de estadia resultantes de uma captação

Devido ao facto de existirem vários pontos de estadia próximos uns dos outros, fazendo referência a um mesmo lugar, é efetuado um agrupamento dos mesmos. O algoritmo de agrupamento usado é o *DJ-Cluster* introduzido na secção 2.3.2, sendo o resultado final todos os locais que se pretendem identificar. Dado que a única informação disponível relativamente aos locais são as suas coordenadas e o período de estadia dos mesmos, é necessário realizar um procedimento adicional para extrair outra informação relevante tal como a sua morada ou o tipo de estabelecimento. Usando os serviços disponibilizados pelo *Google Maps* e pelo *OpenStreetMap*, e recorrendo às técnicas de *reverse geocoding* introduzidas na secção 2.5, é possível complementar a informação anterior associada a cada lugar. A partir do tipo da informação proveniente destes serviços é possível inferir logo algumas atividades devido ao tipo de local que é encontrado, como por exemplo a atividade "Abastecer".

3.3 Identificação de atividades

Após a extração dos locais, é iniciado o processo de extração de atividades, com principal foco nas atividades "Marcar/Desmarcar parcela" e "Conduzir". É necessário fazer uma análise ao tipo de movimentação que é praticada entre os pontos de estadia, ou seja, fazer uma medição das velocidades dos movimentos entre os pontos, por forma a perceber se a pessoa se está a movimentar entre os dois lugares a pé ou a conduzir (Figura 3.4). No entanto, as captações através de sensores *GPS* são muito suscetíveis a erros, o que levou a que não fosse possível determinar o tipo de movimento trabalhando diretamente com os pontos das captações.

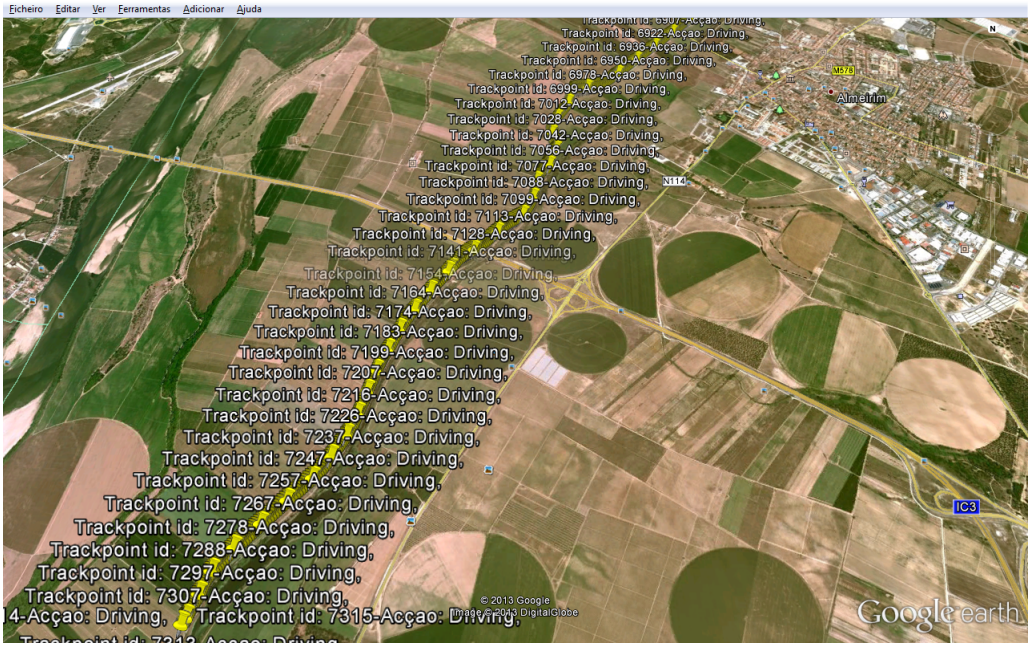


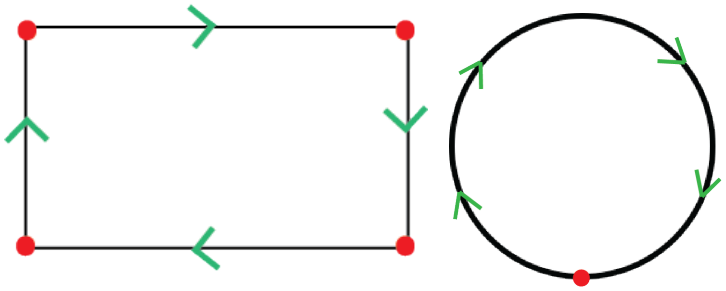
Figura 3.4: Movimentação entre 2 pontos de estadia

Deste modo, para lidar com o erro, foi decidido dividir os dados em janelas. Uma janela, tal como apresentado na figura 3.5, consiste num intervalo de pontos captados seguidamente com uma dimensão fixa, e ao qual está associada uma coordenada média que o representa. Dividindo os pontos iniciais da captação em intervalos iguais leva a que, em vez da velocidade entre os pontos de estadia ser determinada através das coordenadas dos pontos da captação, esta medição passe a ser feita usando a coordenada média de cada janela. Após dividir os dados em janelas e de ser feita a classificação do tipo de movimento praticado entre os vários períodos, é contabilizada a ação de movimentação que mais se exerceu, podendo ser ela conduzir, andar, ou estar parado. Sendo o objetivo localizar troços em que o utilizador está a fazer marcações, são isolados os períodos em que a pessoa começa por conduzir até um ponto de estadia, se desloca a pé em vários períodos e, por fim, volta a conduzir. Aplicando esta metodologia, é possível isolar os momentos em que a pessoa se esteve a deslocar a uma velocidade baixa ou quase nula.

Figura 3.5: Conceito de janela

Após restringir os períodos de análise aos que são pretendidos e ser novamente aplicado o algoritmo da divisão em janelas, é realizada novamente uma classificação. Esta nova classificação tem como objetivo isolar períodos em que o tipo de movimento é andar

ou em que a velocidade de movimentação é quase nula. Assim, contrariamente à classificação anterior, em que os períodos poderiam incluir situações de condução, nestes novos troços as ações são exclusivamente andar ou apresentar uma velocidade baixa/nula. Estando isolados e classificados os períodos que reúnem as condições para as descobertas das atividades, são agrupadas todas as janelas cuja classificação resulta em apresentar uma velocidade baixa/nula, e através da ligação das coordenadas médias das mesmas é identificado o padrão, que consiste em parar num sítio inicial, dar a volta à parcela e por fim voltar ao mesmo sítio. Através deste método, é possível identificar este tipo de sequências, possibilitando assim extrair a área de uma determinada parcela e o sentido de movimento. Conjugando toda a informação que pode ser inferida a partir das sequências, é possível então identificar as atividades pois os padrões reconhecidos nas sequências são específicos e foram caracterizados e desenvolvidos no contexto da dissertação. Na figura 3.6 encontra-se o padrão, já referido anteriormente, que explica a forma de marcação das parcelas regulares e circulares. A desmarcação das mesmas é similar, mas o contorno deve ser efetuado no sentido contrário ao dos ponteiros do relógio.



(a) Indicações para marcação de parcelas regulares

(b) Indicações para marcação de parcelas circulares

Figura 3.6: Tipos de marcações

A visualização de toda a informação é disponibilizada através de uma interface que permite observar o resultado de todo o processo de extração da informação. A partir dela é possível, para além de visualizar, corrigir/complementar a informação extraída.

3.4 Conclusão

Ao longo deste capítulo foram descritos os princípios subjacentes à abordagem que permitiu criar a solução que resolve os objetivos desta dissertação. Toda a solução seguida tem por base um conjunto de decisões e algoritmia que serão descritos e apresentados ao longo dos próximos capítulos, começando pela modelação do sistema.

4

Modelação

Neste capítulo serão descritas as decisões relativas à modelação da solução implementada. Inicialmente será apresentado o diagrama que representa a organização do sistema e, por fim, o diagrama entidade relação que suporta o protótipo desenvolvido.

4.1 Organização do sistema

Na figura 4.1 encontra-se o diagrama que agrupa os vários componentes do sistema desenvolvido. Tal como se pode observar no diagrama, é a camada da aplicação móvel que origina os dados *GPS* das captações, produzindo os ficheiros *GPX*, que servem de informação base às extrações e inferências feitas posteriormente na camada da Aplicação *Web*. A descrição do processo de aquisição e carregamento dos dados está descrito nas secções 5.1 e 5.2 respetivamente. A camada da Aplicação *Web* é constituída por 3 camadas principais: a camada de base de dados, a camada lógica da aplicação e por último, a interface de visualização.

Toda a informação que é produzida pela aplicação é armazenada na camada de base de dados. Esta camada é constituída pelo sistema de base de dados *PostgreSQL* e pela sua extensão o *PostGIS* que permite lidar com os dados espaciais produzidos pela aplicação.

A camada lógica é a responsável por extrair e inferir toda a informação relacionada com a descoberta de lugares e extração atividades. Para o efeito existem 3 componentes principais e uma camada auxiliar. A camada auxiliar, ou seja a camada dos serviços de *GeoCoding*, permite complementar os resultados que são obtidos pelos algoritmos presentes na solução desenvolvida com informação proveniente de vários serviços *online*. O componente de extração de locais é descrito na secção 5.3 e permite obter quais os locais visitados pelo utilizador, sendo que algumas atividades, como por exemplo, *Abastecer*

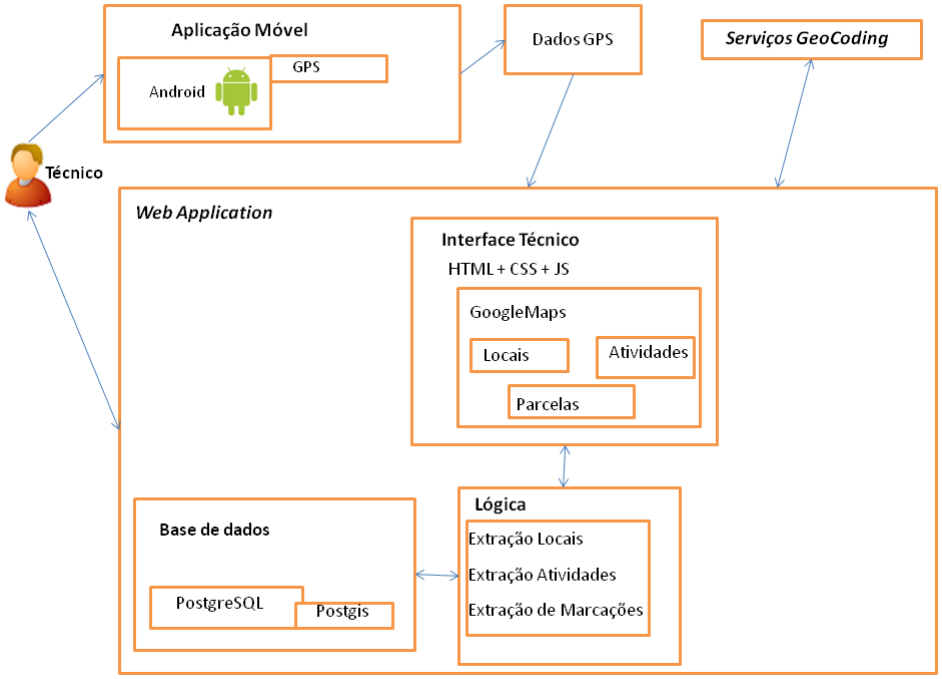


Figura 4.1: Componentes do sistema

podem ser logo inferidas pelo tipo de local em que se encontra.

Para além do componente responsável pela identificação de lugares, existem mais 2 responsáveis pela inferência de atividades. É através destes componentes que é possível extrair e inferir as atividades. A descrição da implementação dos vários mecanismos presentes na solução desenvolvida, relacionados com a extração de atividades está presente na secção 5.4.

Por último a camada de interface é a responsável por permitir visualizar toda a informação, permitir a interação entre o utilizador e a solução desenvolvida, por exemplo para correção de algum lugar mal classificado, e por fim permitir a visualização do relatório final ao técnico.

De referenciar ainda que toda a aplicação *web* e as unidades de processamento foram desenvolvidas utilizando a linguagem Java, num ambiente de computador *desktop*. Neste capítulo apenas se apresentam os vários módulos do sistema, deixando os resultados experimentais e as análises para serem discutidos no capítulo 6.

4.2 Diagrama entidade relação

A visão geral do modelo de dados relacional pode ser observada na figura 4.2.

O modelo permite, em primeiro lugar, suporte total e compatibilidade com o ficheiro *GPX* que se obtém a partir da aplicação móvel. São assim suportados *waypoints*, *staypoints* (pontos de estadia) e *trackpoints*. Estando envolvidos dados espaciais existe uma

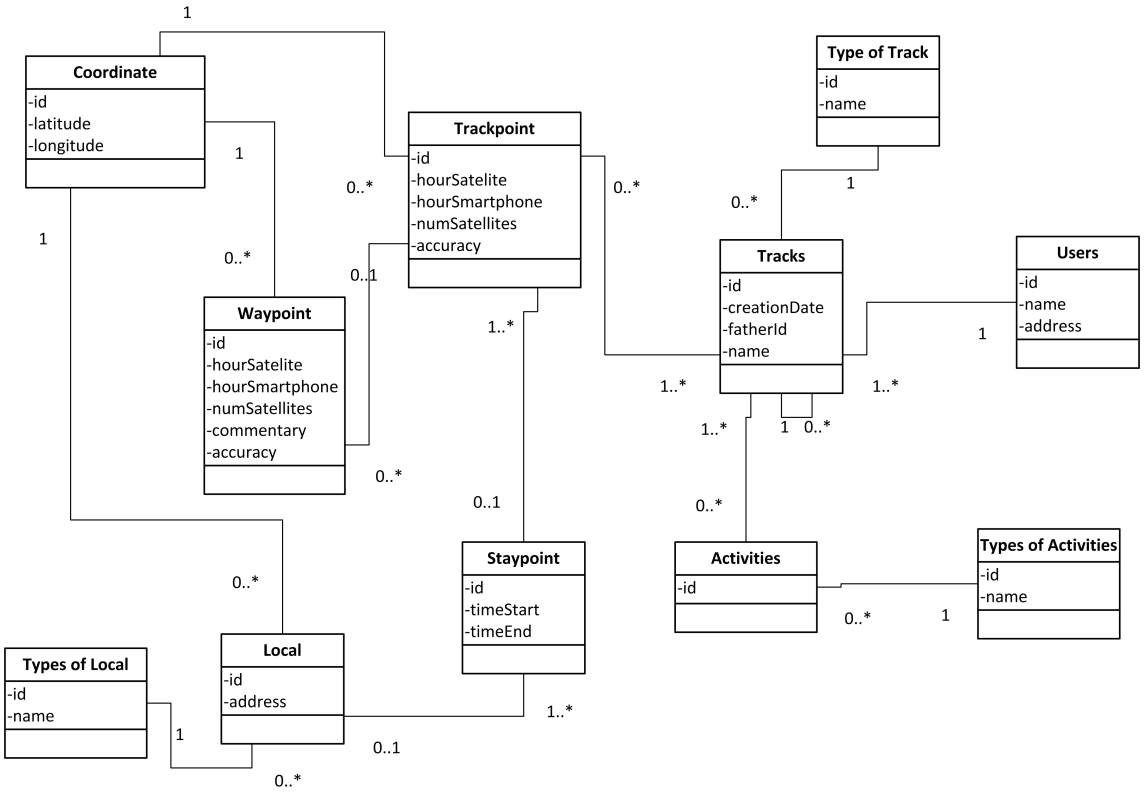


Figura 4.2: Diagrama entidade-relação

tabela que guarda todas as coordenadas dos pontos, a tabela *Coordinate*. Existe também uma tabela que permite guardar toda a informação associada aos utilizadores da aplicação. Por forma a manter a normalização da base de dados foram criadas tabelas que representam os vários tipos possíveis tanto de atividades a detetar, de lugares ou tipo de *tracks*. São elas respetivamente as tabelas *Types of activities*, *Types of places* e *Types of tracks*. Todas as atividades extraídas pela aplicação serão guardadas na tabela *Activities*.

Após ser transferida a informação da captação para a base de dados, a primeira informação a registar nesta é uma *track* com o tipo "*track* original". Esta *track* contém a informação no seu estado puro, servindo de base para todas as inferências e resultados que possam vir a ser obtidos pela aplicação. Todos os pontos de estadia e pontos de interesse dizem sempre respeito à *track* original guardada na base de dados.

No modelo definido e representado na figura 4.2, a cada atividade está associada uma *track* do tipo "*track* de atividade". Este tipo de *tracks* é criado pois cada atividade nova define uma nova sequência de pontos, que representam o percurso que a demarca. No entanto, como todas as novas *tracks* resultam de informação proveniente de uma *track* original, que é a que resulta da captação *GPS* original, existe sempre uma relação com a *track* original correspondente, daí a ligação da tabela *track* com ela mesma.

4.3 Conclusão

Ao longo deste capítulo foram apresentadas e descritas as decisões tomadas relacionadas com todo o planeamento da solução desenvolvida. Foram apresentados os vários componentes que compõem o sistema e o modelo da base de dados que permite armazenar toda a informação. As razões que justificam as decisões presentes neste capítulo são, por um lado, obter um bom funcionamento da aplicação e por outro permitir que haja uma boa base para os vários mecanismos de algoritmia que permitiram atingir os objetivos e que são descritos no próximo capítulo.

5

Implementação

Ao longo deste capítulo discute-se toda a implementação. Na secção 5.1 são apresentadas todas as decisões tomadas relativas à aplicação móvel que permite angariar os dados. Por outro lado, em 5.2, é descrita a abordagem tomada para o carregamento dos dados na aplicação *Desktop*. As últimas 3 secções descrevem as opções tomadas para o desenvolvimento dos métodos de extração de lugares, identificação de atividades e por fim o desenvolvimento da parte gráfica que permite observar e corrigir, caso necessário, os resultados. Toda a parametrização apresentada neste capítulo é a que melhor serve a solução desenvolvida estando toda a avaliação da mesma presente no capítulo 6.

5.1 Aplicação Móvel

Tal como foi descrito no capítulo 3, toda a angariação de dados é efetuada com recurso a uma aplicação *Android*. Para o efeito é utilizada uma versão alterada da aplicação *OSMTracker* [1]. O *OSMTracker* é uma aplicação que permite fazer captações *GPS*, as quais são chamadas de *tracks*/pistas, e que no fim permite que estas sejam exportadas para um ficheiro *GPX*. Devido à grande comunidade, ao facto de o código fonte ser livre e à vasta documentação disponível, facilitando a sua customização, a aplicação foi escolhida como sendo a ideal para adaptar à situação desta dissertação.

Foram realizadas duas alterações na aplicação inicial, uma na *interface* e outra no tipo de informação associada a cada ponto captado. Por defeito, a aplicação fornece uma *interface* que possibilita aos utilizadores identificar, caso queiram, qual o tipo de atividade que estão a praticar no momento. No entanto, como algumas atividades saem fora do âmbito desta dissertação, o aspeto da aplicação foi alterado de forma a que as únicas

[1] http://wiki.openstreetmap.org/wiki/OSMtracker

atividades que utilizador pode identificar são as atividades no qual esta dissertação se foca.

A segunda e última alteração permite a associação de dois tipos de horários a cada ponto captado pela aplicação: o horário fornecido pelo *smartphone* e o horário obtido através de *GPS*. Inicialmente, a aplicação limitava a escolha do sistema horário em vigor a uma destas opções. Assim, modificando o código fonte da aplicação, a base de dados que a suporta passa a conter mais um campo possibilitando, para quaisquer pontos, o registo dos dois tempos em simultâneo. As principais razões para manter os dois tipos de horário são por um lado, a necessidade de garantir a uniformidade da hora entre dispositivos diferentes, usando a hora do satélite e por outro possibilitar o cruzamento de informação interna do *smartphone*, como por exemplo o registo de chamadas, com outras atividades identificadas ou recolhidas, através da hora do *smartphone*. Um exemplo seria uma situação em que o técnico estivesse a conduzir e necessitasse de realizar uma chamada. O técnico pararia de conduzir e seria possível associar os *trackpoints* seguintes à atividade "Falar ao telefone". Utilizando a modificação realizada na aplicação móvel, dado que sabemos a hora do *smartphone*, seria possível associar para além dos *trackpoints* à atividade, complementar com a informação da chamada guardada no registo do próprio dispositivo.

5.2 Carregamento de dados

Apesar da aplicação móvel guardar a informação relativa às *tracks* internamente numa base de dados *SQLite*, devido às restrições do *Android* em isolar o ficheiro de base de dados das aplicações, é difícil ter acesso a essa informação. Isto leva a que a captação tenha de ser exportada para um ficheiro *GPX*, tal como disponibiliza a aplicação *OSMTracker*. Antes de toda a informação ser inserida na base de dados *PostgreSQL* usando a aplicação *Java* desenvolvida para esta dissertação, é necessário transferir o conteúdo do *GPX* para uma nova base de dados *SQLite* através de um mecanismo automático auxiliar. O mecanismo desenvolvido faz parte da aplicação final, sendo ele um módulo que auxilia no pré-tratamento da informação.

Foi criado um processo de automatização baseado num ficheiro *shell script*, que por sua vez faz uso de 3 folhas *XSLT* para automatizar todo o processo de carregamento da informação da captação novamente para uma base de dados *SQLite*. A escolha do uso de folhas *XSLT* está relacionado com o facto do ficheiro *GPX* ser acima de tudo um ficheiro *XML* e, usando folhas *XSLT*, é possível extrair e transformar a informação para um formato passível de ser lido. Foram criados 3 ficheiros *XSLT* responsáveis por extrair informação sobre as *tracks*, *waypoints* e *trackpoints* respetivamente. A aplicação das 3 folhas *XSLT* resulta em 3 ficheiros *TSV* (*Tab separated values*) capazes de serem lidos pelo *SQLite* e, fazendo uso de funções do sistema de base de dados, a informação é carregada. Assim o ficheiro *shell script* descrito em 5.1 funciona da seguinte forma:

Listing 5.1: Ficheiro *shell script*

```
Input: GPX file with the information about the recorded GPS data
Output: SQLite file with all the information about the recorded GPS tracks,
    waypoints and trackpoints

1. Create new SQLite database
2. Apply the XSLT stylesheets to the initial GPX file
  2.1 Extract tracks
  2.2 Extract trackpoints
  2.3 Extract waypoints
3. Load results from step 2. to the database previously created in 1.
```

Após a execução do processo *shell script*, a informação contida na base de dados *SQLite* é inserida na base de dados *PostgreSQL*. A inserção da informação é realizada usando a aplicação *Java* desenvolvida para esta dissertação utilizando a abstração fornecida pelo *Hibernate*. A opção de carregar a informação primeiro para uma base de dados *SQLite* e só depois para o *PostgreSQL* advém do facto de, em primeiro lugar, o *SQLite* ser o sistema de base dados nativo do *Android* e também do facto de ser um sistema de base de dados minimalista, só com algumas funções, sendo o *PostgreSQL* um sistema de base de dados mais sofisticado, capaz de lidar com dados espaciais.

5.3 Detecção e identificação de lugares

A identificação dos locais visitados pelo utilizador é efetuada em 3 fases distintas: a extração dos pontos de estadia, o agrupamento dos mesmos e por fim é realizado um complemento aos resultados obtidos pelo algoritmo extraindo as informações de serviços de mapas online.

A primeira fase consistiu na implementação do algoritmo descrito em 2.3 para a extração dos pontos de estadia. Fazendo uso do *ORM* que é utilizado e da linguagem *Java*, é possível consultar os dados que estão na base de dados e proceder à aplicação do algoritmo sobre esses mesmos dados. Ou seja, o método implementado recebe como *input* os vários *trackpoints*, que são obtidos por captação *GPS*, e como resultado devolve os vários pontos de estadia identificados. Estes são depois guardados na base de dados, respeitando o modelo representado em 4.2. Para alcançar resultados corretos, foi necessário definir os valores das variáveis parametrizáveis presentes no algoritmo: o *distanceTresh* e o *timeTresh*.

***distanceTresh*:** Indica o raio da distância que deve ser considerado para calcular os pontos de estadia do tipo 2. O valor escolhido foi de 100 metros.

***timeTresh*:** Indica o tempo que é necessário despender numa determinada área para ser considerado ponto de estadia. O valor que permitiu obter melhor precisão de resultados foi de 1 minuto e 30 segundos.

Para realizar a segunda fase, o agrupamento dos pontos de estadia, é utilizado o algoritmo *DJ-Cluster* também apresentado em 2.3. No entanto, foi também detalhada uma alteração ao algoritmo por este não ter em conta o tempo despendido no ponto que iria ser agrupado, tendo só em consideração o tamanho da vizinhança, dada uma determinada distância de procura. Por exemplo, se um determinado ponto tem poucos vizinhos, antes da alteração ao algoritmo o ponto seria descartado. Implementada a alteração que tem em conta a quantidade de tempo que é despendida nesse ponto, este passa a ser incluído como um ponto válido. Ou seja, não tem vizinhos suficientes mas é valido pois foi despendido um tempo considerável nele. Como tal, o algoritmo que está implementado é o *DJ-Cluster* com a alteração proposta mencionada na secção 2.3 deste documento. O agrupamento de locais é executado da mesma forma, com base na densidade, mas depois a decisão do algoritmo passa também pela avaliação da duração despendida, decidindo assim se deve, ou não, ser considerado um local importante.

Para implementar o algoritmo foram criados métodos na aplicação *Java* construída, de modo a que esta suporte também esta fase de agrupamento. Como tal, recorrendo mais uma vez ao *ORM* utilizado, são consultados na base de dados todos os pontos de estadia que foram encontrados na fase anterior e é feito um agrupamento dos mesmos. No final desta fase, todos os pontos de estadia que não reúnem as condições para pertencerem a um grupo são descartados. Tal como no primeiro algoritmo utilizado, também neste é necessária uma parametrização. Os parâmetros definidos são:

***eps*:** Indica o valor do raio do círculo que delimita a área em que vão ser procurados os vizinhos. A distância foi definida em 111 metros.

***minPoints*:** Indica o número mínimo de vizinhos necessários para poder ser considerado um local válido. O valor final deste parâmetro foi 2.

***minDrt*:** O valor que mostrou melhor desempenho foi 15 minutos. Este valor determina o tempo mínimo de permanência.

Caso se pretenda agrupar um ponto de estadia, o algoritmo avalia a inclusão do ponto no resultado final, verificando se para uma determinada área definida por (*eps*), o número de vizinhos do ponto é superior a um número mínimo de vizinhos (*minPoints*) ou, por outro lado, se foi despendida uma determinada duração mínima dada pelo parâmetro (*minDrt*).

Por forma a realizar um agrupamento eficiente foram utilizadas *R-Trees*. Estas estruturas estão preparadas para indexar dados geográficos e permitem posteriormente que sejam feitas pesquisas como por exemplo, encontrar pontos num raio parametrizável a partir de uma localização pré-determinada.

Determinados os grupos nos quais cada ponto de estadia se inclui, esta informação é complementada usando *reverse geocoding*. O *reverse geocoding* é uma técnica, disponível nos vários serviços de mapas, que permite, dada uma coordenada (latitude,longitude) devolver o endereço ou o nome do lugar em causa.

A última tarefa pode também ela ser dividida em duas fases: a fase de *reverse geocoding* e a fase de descoberta de locais importantes próximos. Em primeiro lugar é usado o serviço *Nominatim* do *MapQuest* para determinar qual o endereço real de um determinado ponto. Este tipo de operação pode ser executada usando a *API* fornecida pelo *MapQuest* sendo apenas necessário aceder a um endereço[2], fornecer os parâmetros *lat* e *lon* cujos valores devem ser os da coordenada a avaliar e, por fim, extrair a informação que pretende do ficheiro *JSON* retornado.

Tal como foi inicialmente descrito na secção 1.2, existe uma necessidade de descobrir os lugares onde o utilizador almoça/janta, quais as empresas que visita ou quais as bombas de gasolina em que abastece. Como tal, a última tarefa consiste em interrogar o *Google*, usando a *API* do *Google Maps*, por forma a procurar se há um lugar do tipo mencionado em cima, próximo da localização que se está a avaliar. Para obter essa informação é usado o serviço *nearbysearch* do *Google Maps* que, tal como o *Nominatim*, está acessível por um endereço [3]. Para obter a informação desejada, é necessário fornecer a latitude e a longitude da coordenada que identifica o lugar a avaliar no parâmetro *location*, indicar quais os tipos de local que se pretendem através do parâmetro *types* e por fim qual o raio de procura *radius*. No nosso caso são requisitados os lugares mais próximos do tipo: *restaurant* e *establishment* num raio de 100 metros. A resposta do pedido ao endereço é um ficheiro *JSON* em que os resultados aparecem por ordem de importância dada pela popularidade do local. Caso a condição anterior não se aplique, o processo devolve o lugar mais próximo, do tipo escolhido. As bombas de gasolina são também retornadas através do tipo *establishment*.

Após complementada a informação inferida pelos algoritmos da aplicação com a informação dos dois serviços mencionados, como resultado final, obtemos, para todos os grupos criados, uma correspondência a um lugar físico com um endereço e, se possível, um estabelecimento associado. Existe também outra forma de identificar um novo lugar que tem por base o facto de este já ter sido visitado anteriormente. Ou seja, caso a localização de um novo lugar seja muito próxima de outro previamente descoberto, não são feitos os pedidos aos serviços de *geocoding*, o novo lugar é imediatamente classificado de igual forma ao antigo. Por exemplo, se numa captação anterior um determinado lugar for classificado como a casa do técnico, caso numa nova captação a coordenada do lugar resultante seja muito próxima da casa do técnico, o lugar será classificado como sendo o mesmo local.

5.4 Identificação de Atividades

A extração das várias atividades a partir da captação inicial é conduzida em várias fases. Ao longo das várias fases o resultado de cada uma é aproveitado para a fase seguinte,

[2] http://open.mapquestapi.com/nominatim/v1/reverse.php?format=json

[3] https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=latitude,longitude&sensor=true&language
d3YhgvPKu3MLs&rankby=prominence&radius=radius&types=TYPES

sendo a informação restringida até estar só a necessária para extrair as atividades pretendidas. A identificação foi dividida em 3 fases. A primeira consiste na análise e identificação do tipo de movimento mais exercido entre os vários pontos de estadia. Os movimentos podem ser classificados como períodos de *driving*, *walking* ou *almostStoppedOrStopped*. A segunda fase consiste na extração das várias sequências nas quais é possível identificar as várias atividades. As várias sequências são originadas a partir do reconhecimento de um padrão, tendo por base o resultado do tipo de movimento registado nos períodos entre os pontos de estadia. Por fim, a terceira fase consiste na extração das várias atividades a partir das sequências que resultam da segunda fase, especialmente a marcação e desmarcação de parcelas.

5.4.1 Primeira fase - Análise ao tipo de movimento exercido entre os pontos de estadia

A primeira fase consiste em realizar uma análise ao tipo de movimento que é exercido entre os vários pontos de estadia determinados durante a identificação dos lugares. A realização da análise prende-se com a necessidade de identificar qual a velocidade a que o técnico se está a movimentar entre os mesmos, isto porque algumas atividades como "Conduzir" são praticadas a uma determinada velocidade na deslocação entre os vários pontos de estadia enquanto que, por exemplo, as marcações/desmarcações são exercidas nos períodos de baixa velocidade que são delimitados por pontos de estadia que são precedidos e sucedidos por períodos de condução. No entanto, dada a suscetibilidade dos dados *GPS* a erros, após identificar os *trackpoints* que estão entre cada dois pontos de estadia é necessário dividi-los em janelas de movimento. A divisão dos dados em janelas implica que em vez de se lidar com um *trackpoint* se lide com a coordenada média dos vários intervalos de *trackpoints* (Figura 5.1). Ou seja, consegue-se minimizar o erro da captação original. A primeira fase da extração e identificação de atividades termina com a identificação do tipo de movimentação mais exercida entre cada ponto de estadia para posteriormente se poderem identificar as atividades pretendidas.

Figura 5.1: Conceito de janela

O pseudo-código do algoritmo que permite dividir os dados em janelas está apresentado em 5.2.

Listing 5.2: Algoritmo de divisão em janelas

```
1 Input: List of trackpoints ( trackpointsList )
2 Output: List of Windows (listOfWindows)
3
4 for i:=0 to trackpointsList.size do
```

```
  j:=0, currPos:=i, Window w
    while j<windowSize do
      add trackpointsList[currPos] to w;
      j++;
      curPos++;
    end

    set time spent on w
    set mean coordinate of w
    set mean velocity of w

    add w to listOfWindows
end

return listOfWindows
```

O único parâmetro do algoritmo é o *windowSize* que permite especificar qual a dimensão da janela. O valor definido para este parâmetro foi de 60. O algoritmo itera sobre os vários *trackpoints* e retorna os mesmos divididos em janelas. O algoritmo recebe um conjunto de *trackpoints* e devolve uma lista com todas as janelas, cada uma com *windowSize trackpoints*. As várias janelas são sobrepostas, isto quer dizer que o último *trackpoint* de uma determinada janela faz parte da próxima como sendo o *trackpoint* no qual tem início a janela. Tendo por base todos os *windowSize trackpoints* atribuídos a uma janela, é calculado o tempo despendido nela, a sua coordenada média e a sua velocidade. A coordenada média é obtida calculando a média das coordenadas da janela. Por outro lado, a velocidade é calculada usando as várias coordenadas contidas e o tempo despendido na janela.

O resultado do algoritmo anterior serve de *input* ao algoritmo, desenvolvido propositadamente para esta dissertação, que permite agrupar as várias janelas pela variação de velocidade praticada nas mesmas. O algoritmo, cujo pseudo-código está em 5.3, identifica os vários tipos de movimentação, entre cada ponto de estadia. O algoritmo recebe as várias janelas como *input* e retorna uma lista com todas as janelas agrupadas e classificadas consoante a sua velocidade. O algoritmo usa também um outro como auxiliar, criado também propositadamente, que permite tratar dos casos em que a velocidade é baixa ou quase nula. O pseudo-código do algoritmo auxiliar está em 5.4. Neste caso, o algoritmo recebe uma lista de janelas que garantidamente vão ser agrupadas e classificadas como *walking* ou *almostStoppedOrStopped* que, por sua vez, vão ser adicionadas ao algoritmo principal descrito em 5.3.

Listing 5.3: Algoritmo de identificação do tipo de movimento

```
Input: List of Windows (listOfWindows)
Output: List of pairs <action,list of Windows> (resultList)

//create result list resultList
minor:=false, merged:=false, List<Window> currList

```

```
//initial case
//classify the first element
  if (listOfWindows.get(1).midVelocity ≤ velocityParameter)
    minor :=true

  create new currList
  currList.add(listOfWindows.get(0))
  currList.add(listOfWindows.get(1))

//iterate over all the elements except de first one (which was already treated)
for j:=i+1 to listOfWindows.size do
  //condition to classify as driving classification
  if(listOfWindows.get(j).midVelocity <= velocityParameter && !minor) then
    minor:=true

    if(!merged) then
      create new pair newPair
      newPair := <driving,currList>
      add newPair to resultList

    else
      resultList.get(resultList.size-1).append(currList)
    end

    create new currList
  end

  //condition to classify as walking/stopped or almostStopped
  if(listOfWindows.get(j).midVelocity > velocityParameter && minor) then
    minor:=false

    //condition that verifies if it was not just a momentary reduction of speed
    if(currList.size ≥ numMinWindows || resultList == 0) then
      create new List of pairs <action,list of windows> newListOfPairs

      //call to the auxiliary method
      newListOfPairs = determineIfWalkingOrAlmostStopped(currList)

      //add all the results
      for p ∈ newListOfPairs do
        resultList.add(p)
      end

      create new currList

    else
        merged:=true
    end
  end
```

```
57    resultList.add(listOfWindows.get(j))
58  end
59
60  treat last currList as the other ones
61
62  return resultList
```

Existem duas variáveis parametrizáveis nos algoritmos, *velocityParameter* e *numMinWindows*.

***velocityParameter*:** Este parâmetro delimita a velocidade a partir da qual se pode classificar um movimento como estar a conduzir (*driving*) ou a andar a uma velocidade baixa (*walking*) ou muito baixa/nula (*almostStoppedOrStopped*). O valor escolhido é 2.75 metros por segundo (aproximadamente 10 quilómetros por hora).

***numMinWindows*:** Este valor representa o número mínimo de janelas necessárias para criar um novo conjunto de janelas com uma classificação diferente da atual. Por exemplo, consideremos que num determinado ponto da execução do algoritmo, a classificação, para um determinado conjunto de janelas, é *driving*. No entanto as próximas janelas representam um momento em que o utilizador teve de reduzir a velocidade durante algum tempo, devido a, por exemplo, ter de parar num sinal *STOP* ou num semáforo. Estes períodos levam a que a velocidade média nas janelas diminua. Isto faria com que o algoritmo adicionasse o conjunto de janelas atual ao resultado final como *driving* e iniciasse um novo ciclo em que a classificação possível poderia ser *walking* ou *almostStoppedOrStopped*. Usando este parâmetro é possível lidar com esse tipo de situações e diminuir o erro de classificação. O valor escolhido é de 30 janelas.

O algoritmo descrito em 5.3 percorre todas as janelas e compara a cada iteração a velocidade média de cada janela com o parâmetro *velocityParameter*. Inicia com a verificação da velocidade média da primeira janela, colocando a classificação como estar ou não a conduzir. O tipo de classificação é registado pela variável *minor*. O algoritmo itera e vai adicionando janelas à lista temporária ativa no momento, até encontrar uma velocidade que o faça entrar numa das condições. Existem duas condições no algoritmo que o podem fazer mudar de estado: ou, no estado atual, o algoritmo encontra-se num período de velocidade elevada e encontrou uma velocidade baixa ou a situação contrária. Como já foi explicado, devido ao parâmetro *numMinWindows* é possível diminuir a sensibilidade do algoritmo e evitar falsos positivos, não sendo contabilizadas situações de paragem ou, de redução momentânea. Na figura 5.2 pode-se observar parte de um conjunto de janelas que está a ser classificado como *driving* (condução). É possível observar que, apesar de a determinada altura a velocidade ser inferior ao parâmetro de 2.75 metros por segundo, a classificação final não se altera pois o período de tempo de deslocação a baixa velocidade não foi o suficiente para mudar o tipo de classificação.

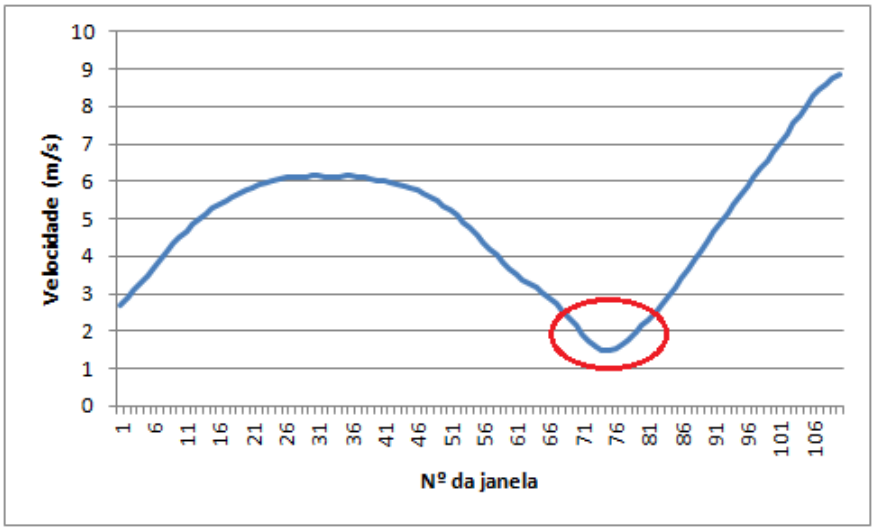


Figura 5.2: Período de condução com redução temporária de velocidade

No final, tal como descrito anteriormente, a classificação para um conjunto de janelas, pode ser *driving*, *walking* ou *almostStoppedOrStopped*. A primeira é inferida diretamente se a velocidade é superior a *velocityParameter*. No entanto, as outras duas classificações são determinadas recorrendo a um método auxiliar, também criado especificamente para esta dissertação, cujo pseudo-código se encontra em 5.4. É feita uma aproximação dos dados recebidos no *input* a uma distribuição normal, possibilitando obter a média das velocidades das várias janelas, o desvio padrão e o valor que determina se a classificação deve ser *walking* ou *almostStoppedOrStopped*. O algoritmo itera sobre as várias janelas e vai criando grupos começando por determinar, logo ao início, se estamos numa situação de *walking* ou *almostStoppedOrStopped*. A cada iteração, consoante o resultado da comparação da velocidade da janela atual com a expressão 5.1, o algoritmo ou adiciona ao conjunto atual que está na lista temporária ou adiciona a lista temporária ao resultado, criando uma nova lista. A criação de uma nova lista representa a alteração de uma situação *walking* para uma *almostStoppedOrStopped* ou o inverso. Um conjunto de janelas é considerado *walking* se todas as velocidades médias das janelas se encontrarem acima do valor obtido na linha 6 do algoritmo (5.1). O algoritmo é executado tantas vezes quanto o número de períodos entre pontos de estadia. A aplicação do algoritmo a cada período resulta num conjunto de classificações *driving*, *walking* ou *almostStoppedOrStopped*.

A expressão 5.1 foi obtida mediante um conjunto de observações efetuado ao longo da dissertação com as várias captações. Pretende-se identificar como *almostStoppedOrStopped* todas as velocidades que, dada a distribuição normal, têm apenas 5% de confiança. Assim, é utilizado o quartil $z_{0.975}$ e fazendo a subtração entre a média das velocidades e os 5%, obtém-se o valor da velocidade que determina se estamos numa situação *walking* ou *almostStoppedOrStopped*.

$$valueThatDecides := |(average - (1.96 \times standardDeviation)| \quad (5.1)$$

Listing 5.4: Algoritmo auxiliar de identificação do tipo de movimento a baixa velocidade

```
1  Input: List of Windows (listOfWindows)
2  Output: List of pairs <action,list of Windows> (resultList)
3
4  average:= average of sample
5  standardDeviation:= standard deviation of sample
6  valueThatDecides := |(average - ( 1.96*standardDeviation))|
7  stopped:=false
8
9  create result list resultList
10 create new currList
11
12 //initial case
13 //verify if the first value found
14  if (listOfWindows.get(0).midVelocity ≤ valueThatDecides)
15     stopped:=true
16
17   //add to the temporary list
18   currList.add(listOfWindows.get(0))
19
20   //iterate over the rest of the values
21   while(iterator of listOfWindows.hasNext) do
22     Window w := listOfWindows.nextWindow
23
24     //condition to classify as stopped
25     if((w.midVelocity ≤ valueThatDecides) && !stopped) then
26       stopped :=true
27
28       create new pair newPair
29       newPair := <walking,currList>
30       add newPair to resultList
31
32       create new currList
33     end
34
35     //condition to classify as walking
36     if((w.midVelocity > valueThatDecides) && stopped) then
37       stopped :=false
38
39       create new pair newPair
40       newPair := <almostStoppedOrStopped,currList>
41       add newPair to resultList
42
43       create new currList
44     end
45
46     add w to currList
```

```
47  end
48
49  create new pair newPair
50
51  //treat the last case
52  if(stopped) then
53    newPair := <almostStoppedOrStopped,currList>
54  else
55    newPair := <walking,currList>
56  end
57
58  add newPair to resultList
59
60  return resultList
```

Após obter todas as classificações de cada período, é feita uma contagem com o objetivo de obter o número de vezes que aparece determinada classificação. No final, atribuiu-se um tipo de classificação a todo o período, tendo por base a quantidade de vezes que esse tipo de classificação é encontrado. Ou seja, se entre dois pontos de estadia são encontradas 3 classificações de *driving* e uma de *walking*, a classificação geral do período é de *driving*. O resultado serve como ponto de partida para a próxima fase, que tem como objetivo isolar as sequências onde serão identificadas algumas das várias atividades a extrair.

5.4.2 Segunda fase - Obtenção das sequências nas quais vão ser identificadas as atividades

Determinada a ação mais praticada no período que decorre entre cada ponto de estadia, a segunda fase inicia-se com a aplicação do algoritmo que permite obter os períodos exemplificados na figura 5.3. Dado que as atividades mais relevantes decorrem em períodos de não condução, o objetivo é isolar os períodos delimitados por pontos de estadia cujos períodos que os antecedem e os sucedem são períodos de condução. Para tal é usado o algoritmo cujo pseudo-código se encontra em 5.5.

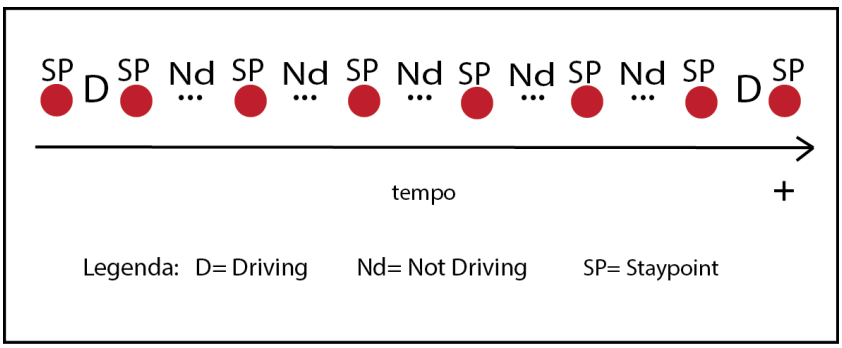


Figura 5.3: Padrão de períodos a isolar

Listing 5.5: Algoritmo de identificação de possíveis períodos de atividades

```
Input: List of Pairs <number of staypoint,action> (listOfPairStaypointIDAction)
Output: List of Lists of Trackpoints (resultList)

create result list resultList

create List<Trackpoint> currList

//starts iterating over the array
for i:=0 to listOfPairStaypointIDAction.size do
  lastIndexOfInterval:=-1, startingIndexOfInterval:=-1

  //if is a case of driving
  if(listOfPairStaypointIDAction.get(i).second == driving) then

    //iterates until a new driving situation appears
    for j:=i+1 to listOfPairStaypointIDAction.size do
      if(listOfPairStaypointIDAction.get(i).second == driving) then
        if(listOfPairStaypointIDAction.get(j).getFirst - listOfPairStaypointIDAction.
            get(j-1).getFirst > 1) && lastIndexOfInterval>0) then
          lastIndexOfInterval := listOfPairStaypointIDAction.get(j-1).getFirst+1
        end
        i:=j-1
        break
      else
        lastIndexOfInterval:=listOfPairStaypointIDAction.get(j).getFirst;
        startingIndexOfInterval:=i;
      end
    end

    //gets all the trackpoints in the interval between the two driving periods
    if(lastIndexOfInterval>0) then
      Staypoints staypoints1 := get Staypoint from database of
          listOfPairStaypointIDAction.get(startingIndexOfInterval).getFirst
      Staypoints staypoints2 := get Staypoint from database of lastIndexOfInterval

      currList := get all trackpoints between staypoint1 and staypoint2

      atTown:=false
      atTown := verify if at least one of the two Staypoints is out of town

      if(atTown)
        add currList to resultList
      end
    end
  end
end

return resultList
```

O algoritmo itera sobre uma lista de pares que é recebida como *input* e que contém, para todos os pontos de estadia, qual a ação mais praticada até cada um. Ou seja, o ponto de estadia 1 tem associada a ação mais praticada entre o ponto de estadia 0 e o ponto de estadia 1. O algoritmo itera até encontrar um par cuja ação é a de *driving* (conduzir). A posição encontrada é assinalada como o ponto de estadia inicial e a iteração continua, num ciclo interno, até voltar a encontrar um período de *driving*. Garantidamente que, no final, todos os períodos retornados são os que permitem obter algumas das atividades efetuadas durante a captação que está a ser avaliada. Ou seja, são os períodos não *driving*. Antes de um determinado período ser adicionado à lista de resultados, o algoritmo verifica usando o serviço *Geocode* da Google se, pelo menos, um dos pontos de estadia está localizado fora de uma localidade. Esta verificação é efetuada pois as atividades que se pretendem identificar, reconhecendo os padrões nas captações, são atividades efetuadas no campo. Como tal, criando este teste, é possível restringir os resultados somente aos períodos que interessam. Por outras palavras, os períodos sobre os quais são aplicados os algoritmos de extração de atividades, incluem somente os trajetos localidade-campo, campo-localidade, campo-campo.

Os períodos obtidos são divididos em janelas usando o algoritmo cujo pseudo-código se encontra em 5.2 e submetidos ao algoritmo de classificação de janelas, descrito em 5.3. Como os novos períodos selecionados se encontram entre dois períodos de condução, quando aplicados ao algoritmo só são encontrados períodos não *driving*. Neste caso, foi realizada uma alteração aos valores dos parâmetros presentes nos dois algoritmos. *numMinWindows* foi alterado para 5 e, o tamanho da janela, *windowSize*, foi alterado para 10. A submissão dos dados obtidos aos algoritmos 5.2 e 5.3, bem como a alteração dos parâmetros, prende-se com a necessidade de fazer uma pesquisa nos dados obtendo o máximo de granularidade possível. Como os períodos de *driving* foram excluídos com a aplicação do algoritmo descrito em 5.5, o resultado inclui apenas períodos de *walking* e *almostStoppedOrStopped*.

Tal como foi indicado aos técnicos, quando quisessem realizar alguma marcação ou desmarcação, deveriam circundar a área, começando e acabando no mesmo sítio, parando no início e no final. Os dados de entrada para o algoritmo que permite obter as várias sequências, descrito em 5.6, são obtidos através do cálculo da coordenada média das várias janelas que foram classificadas como *almostStoppedOrStopped*, tendo obrigatoriamente um período de *walking* entre elas. Por exemplo, se um técnico parar num determinado lugar, andar e voltar a parar noutro, espera-se que o resultado seja um conjunto de janelas a *almostStoppedOrStopped*, seguido de um conjunto de janelas classificadas como *walking* e terminando novamente com um conjunto de janelas classificado como *almostStoppedOrStopped*. O resultado são 3 coordenadas médias, cada uma correspondente a cada período encontrado. No entanto, o algoritmo só recebe, como entrada, as coordenadas dos períodos *almostStoppedOrStopped*.

No algoritmo descrito em 5.6 existem 4 variáveis parâmetros:

minimumTimeToCountAsBackToSamePlace : Este parâmetro determina o tempo mínimo necessário entre dois *almostStoppedOrStopped* para que se possa considerar uma sequência onde poderá ter acontecido uma atividade. O valor determinado para este parâmetro foi de 90000 milissegundos.

distanceToLookForInBackToSameArea : Como o objetivo é encontrar o padrão "começar e acabar no mesmo lugar", este parâmetro indica a distância máxima a que podem estar dois *almostStoppedOrStopped* para se poder considerar que são o mesmo lugar. Esta distância não é 0 pois a pessoa pode não parar exatamente no mesmo lugar e existe erro. O valor definido para este parâmetro foi de aproximadamente 25 metros.

minimumSizeArea : Este parâmetro permite excluir sequências cuja área seja muito pequena. O valor escolhido foi de 10 metros quadrados.

paramDifference : A variável *paramDifference* permite aplicar uma restrição que, implica que após ser detetada uma sequência, a próxima só poderá ocorrer depois de *paramDifference* milissegundos. O valor definido foi de 5 minutos.

Listing 5.6: Identificação das sequências onde decorrem as atividades

```
1  Input: List of Pair <middleCoordinate,List of <Trackpoint> (
       allWindowsOfStoppedType)
2  Output: List of Lists <Trackpoint> (resultList)
3
4  //iterate over all coordinates that represent almostStopped/Stopped periods
5  for i:=0 to allWindowsOfStoppedType.size do
6
7    //start iterating from the second one
8    for j:=i+1 to allWindowsOfStoppedType.size do
9      areClose := false
10
11     //verify if both coordinates are near each other
12     if distance(allWindowsOfStoppedType.get(i),allWindowsOfStoppedType.get(j))<=
           distanceToLookForInBackToSameArea
13       areClose := true
14
15     //codition if are close
16     if areClose then
17       //verify if already has passed minimumTimeToCountAsBackToSamePlace
             milliseconds
18       if(timeDifferenceBetween(allWindowsOfStoppedType.get(i),
             allWindowsOfStoppedType.get(j)) >= minimumTimeToCountAsBackToSamePlace
             ) then
19         create List<Trackpoint> currList
20         currList := get all trackpoints between (allWindowsOfStoppedType.get(i).
               second.get(0).time and (allWindowsOfStoppedType.get(j).second.get(
               last).time
```

```
21
22          //verify if the area is big enough
23          if calculateArea(currList) >= minimumSizeArea then
24            add currList to resultList
25
26            a:=i+1
27            //condition that moves the index of the external cicle to force a
                  difference of paramDifference between the sequences
28            while(a<allWindowsOfStoppedType.size) do
29                difference := calculateDifference(allWindowsOfStoppedType.get(a).
                      second.get(0).time,allWindowsOfStoppedType.get(i).second.get
                      (0).time)
30                if(difference > paramDifference) then
31                  i=a-1
32                  break
33                else
34                  a++
35                end
36            end
37            break
38          end
39        end
40      end
41    end
42  end
43
44  return removeSuperSequences(resultList)
```

O algoritmo itera sobre a lista de pares que recebe como *input*. Por cada iteração realiza uma outra, num ciclo interior, que começa sempre na posição a seguir à do ciclo exterior. É calculada a distância entre a coordenada associada à posição do ciclo exterior e a coordenada associada à posição do ciclo interior, com o objetivo de verificar se ambas estão, no máximo, a uma distancia de *distanceToLookForInBackToSameArea* metros uma da outra. Na implementação *Java* realizada, foi usada uma estrutura *R-Tree*, dada a sua capacidade de indexar dados geográficos e o facto de estar preparada para realizar este tipo de pesquisas. Após verificar que é uma sequência válida, por respeitar a condição de proximidade, são executados mais alguns testes à sequência antes desta ser adicionada ao resultado final.

O primeiro teste verifica se a sequência tem no mínimo uma duração de *minimumTimeToCountAsBackToSamePlace*. Caso se verifique a veracidade desta condição, a sequência é testada quanto à sua área. Se a sequência perfaz uma área de, no mínimo, *minimumSizeArea*, é considerada válida e a lista corrente, *currList*, é adicionada ao resultado. A área é calculada usando uma das funções fornecidas pelo *PostGIS* que permite, fornecendo uma lista de coordenadas, calcular a área ocupada pela junção das mesmas. No final, o algoritmo retorna todas as sequências válidas obtidas.

Uma característica do algoritmo é que, após obter a primeira sequência válida numa determinada iteração dos ciclos, o *break* da linha 32 força a saída da iteração interior, obrigando a iteração exterior a avançar. Esta característica permite reduzir o número de sequências que são obtidas no final e foi incluída no algoritmo pois, após terem sido feitos alguns testes com várias captações, permitia obter melhores resultados, excluindo sequências que seriam super-sequências das já detetadas anteriormente. No entanto, apesar da alteração, nem todas as super-sequências são totalmente removidas, o que levou à criação do algoritmo descrito em 5.7. O algoritmo é aplicado diretamente sobre os resultados obtidos pelo descrito em 5.6 e remove todas as super-sequências presentes na lista que lhe é fornecida como parâmetro de entrada. A motivação para a remoção das super-sequências é ilustrada na figura 5.4. Na figura 5.4 está representado um conjunto de sequências. A imagem representa uma situação em que o técnico pára o carro ao chegar à propriedade, efetua duas marcações e retorna ao carro. A primeira sequência representa todo o tempo desde que o técnico pára o carro para ir marcar as parcelas até voltar, enquanto que as duas últimas representam as duas marcações. Como é possível observar, a primeira sequência é uma super-sequência da segunda e da terceira, pois começa antes e acaba depois de ambas. Assim, é removida da lista. Este é o típico exemplo do que é a remoção das super-sequências.

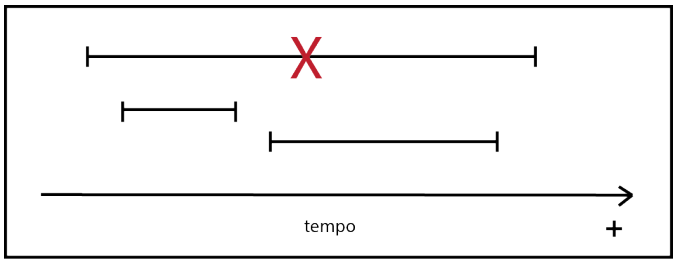


Figura 5.4: Exemplo de remoção de super-sequências

O algoritmo cujo pseudo-código se encontra em 5.7 itera sobre as várias sequências da lista que recebe como *input* e realiza uma verificação com o objetivo de determinar se a sequência na próxima posição do ciclo começa e termina depois da atual. Caso a condição se verifique, a sequência atual é adicionada à lista final. Por outro lado, caso não se verifique, é executado um ciclo interior sobre a lista de resultados final atual com o objetivo de remover todas as sequências que terminam depois da sequência na próxima posição do ciclo exterior. Ou seja, após detetar que a sequência atual do ciclo exterior termina depois da próxima, o algoritmo itera sobre todas as que já foram adicionadas ao resultado final com o objetivo de remover as que terminem depois da sequência que está na próxima posição do ciclo exterior. O algoritmo termina e retorna todas as sequências que representam os vários períodos de tempo em poderão ter ocorrido atividades.

Listing 5.7: Remoção das super-sequências

```
Input: List of Lists <Trackpoint> (originalSequence)
Output: List of Lists <Trackpoint> (resultList)

//iterate over all sequences
for i:=0 to originalSequence.size do
  create List<Trackpoint> currList
  currList := originalSequence.get(i)

  if(i<resultList.size-1) then

    //take the next sequence
    nextTrackpointList := originalSequence.get(i+1)

    //compare the initial time of both
    if(nextTrackpointList.get(0).time >= currList.get(0).time) then
      nextTrackpointLastTrackpointTime := nextTrackpointList.get(nextTrackpointList.size
          ()-1).time

      //compare the final time of both
      if(nextTrackpointLastTrackpointTime <= currList.get(currTrackpointList.size()-1).
          time) then

        //iterate over the sequences that were already on the result
        for j:=resultList.size to 0 do
          //remove the sub-sequences
          if (resultList.get(j).get(resultList.get(j).size()-1).time >=
              nextTrackpointLastTrackpointTime)
            resultList.remove(j)
        end

        //add the sequence
        resultList.add(currList)
        i:=i+1
      else
        resultList.add(currList)
      end
    end
  else
    resultList.add(currList)
  end
end

return resultList
```

5.4.3 Terceira fase - Identificação das várias atividades

A terceira e última fase tem como objetivo a identificação das várias atividades e recebe como parâmetro de entrada as várias sequências retornadas da segunda fase. O primeiro mecanismo que vai ser descrito tem como objetivo detetar o sentido do movimento, ou seja, se este é realizado no sentido dos ponteiros do relógio ou no sentido contrário. Posteriormente são descritos alguns mecanismos que têm como objetivo a identificação da forma da parcela que se está a extrair que, apesar de fazerem sentido teoricamente, nem todos obtiveram os resultados esperados. Por último, é descrita a forma como são identificados os vários tipos de pivôs.

5.4.3.1 Identificação do sentido do movimento

Como foi descrito em 1.2, o técnico deveria contornar as parcelas no sentido dos ponteiros do relógio caso as quisesse marcar ou no sentido oposto caso as quisesse desmarcar. Nesse sentido é aproveitada uma função presente no sistema de base de dados que está a ser usado e o sentido do movimento é determinado usando a função disponibilizada pelo *PostGIS*, *isCounterClockWise*. A função recebe um conjunto de coordenadas como *input* e retorna um valor positivo se as coordenadas estiverem orientadas no sentido contrário ao dos ponteiros do relógio. Através da identificação do sentido do movimento é possível determinar se a atividade foi de marcação ou de desmarcação. Na figura 5.5 está um exemplo do resultado da descoberta do sentido do movimento que, neste caso, resulta numa marcação. É possível observar na figura que o identificador dos *trackpoints* aumenta no sentido dos ponteiros do relógio, o que confirma que se está a marcar uma parcela. Neste caso a sequência é identificada com a atividade "Marcar Parcela" no entanto, caso a deslocação fosse no sentido oposto, seria identificada como "Desmarcar Parcela". A desmarcação de uma parcela corresponde à remoção de uma parte ou totalidade de uma área de terreno.

Figura 5.5: Marcação de parcela

5.4.3.2 Identificação da forma da área detetada

Foram realizadas duas abordagens, para tentar identificar a forma das parcelas detetadas. A primeira abordagem tem por base o conceito de *bearing* ou rumo, enquanto que a segunda faz uso às funções presentes no *PostGIS*. No final, pretende-se identificar se uma parcela é ou não circular, com o objetivo de aquando a identificação da atividade poder complementar se esta foi realizada numa área circular ou não.

Segundo a *Wikipedia* [4] "Rumo é, em náutica, o ângulo que faz a direção seguida por uma nave com uma direção de referência, que geralmente é associada ao Norte magnético ou cartográfico". Assim, a primeira abordagem consiste na realização de histogramas com os vários rumos seguidos numa determinada sequência de *trackpoints*, com o objetivo de determinar quais os ângulos tomados ao longo dela e, no final, contabilizá-los para se poder saber o número de picos que representam os máximos locais. Ou seja, no final, cada ângulo tem um número associado nos histogramas que corresponde ao número de vezes que aparece durante a trajetória da sequência. Por exemplo, se fosse feito um histograma com os vários rumos de uma sequência circular regular, a tendência seria para que todos os ângulos aparecessem pelo menos uma vez no histograma, não havendo máximos locais. Por outro lado, se a sequência a identificar tem uma forma quadrangular perfeita, o histograma deveria acusar 4 picos pois numa estrutura quadrangular há 4 mudanças de direção, logo haveria 4 máximos locais.

O problema de realizar um único histograma, que contabilize o número de vezes que é descoberto um determinado ângulo numa determinada sequência, está relacionado com a intolerância a erros e ao ruído. Como as captações *GPS* contêm sempre algum ruído, haveriam muitos ângulos a aparecer contabilizados devido à instabilidade,

[4] http://pt.wikipedia.org/wiki/Rumo_(n%C3%A1utica)

por exemplo do sinal ou da irregularidade do terreno impossibilitando um rumo linear, durante a captação. Na figura 5.6 está representado o histograma dos rumos seguidos na marcação da área representada na figura 5.5. A barra vertical representa o número de vezes que o ângulo representado na barra horizontal foi contabilizado. Como a área é quadrangular, deveria só haver 4 picos. No entanto é possível observar que existem mais do que 4 picos no histograma.

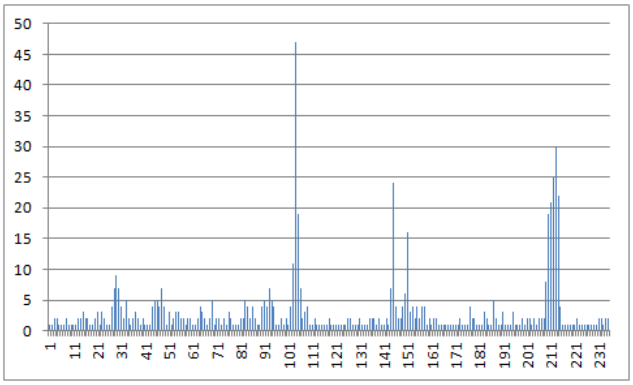


Figura 5.6: Histograma de uma área quadrangular

Desta forma, foram desenvolvidos métodos que, apesar de não permitirem identificar a forma da sequência, possibilitam saber o número de picos e fazer uma análise que permite afirmar, grande parte das vezes, se é ou não circular, sendo capazes de lidar com o ruído dos dados. A metodologia funciona da seguinte forma: recebendo uma sequência de *trackpoints* como parâmetro, são realizados 3 histogramas, cada um contendo 10 classes, ou seja, cada histograma contém 10 posições. Cada posição no histograma representa uma classe, que descreve um intervalo no qual estão englobados 36 graus. Passa assim a ser possível lidar com vários ângulos num só intervalo. Por exemplo, se o ângulo entre duas coordenadas for de 2º e outro for de 3º, ambos devem estar na mesma classe, e aplicando essa divisão dos dados em 10 classes é possível lidar com este tipo de situação. Os 3 histogramas diferem entre eles nos ângulos que delimitam cada intervalo. Ou seja, subdivide-se o intervalo de cada classe em 3 partes e vai-se avançando. Isto evita que picos consecutivos possam ficar em duas classes. Os histogramas funcionam da seguinte forma:

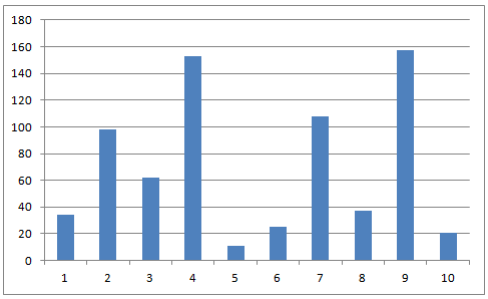
Histograma 1 10 intervalos de 36 graus. O primeiro intervalo é delimitado entre 0º e 35, o segundo por 36 e 71º e assim sucessivamente.

Histograma 2 10 intervalos de 36 graus. O primeiro intervalo é delimitado entre 12º e 47, o segundo por 48 e 83º e assim sucessivamente até ao último intervalo que engloba os ângulos entre 336º e 11º.

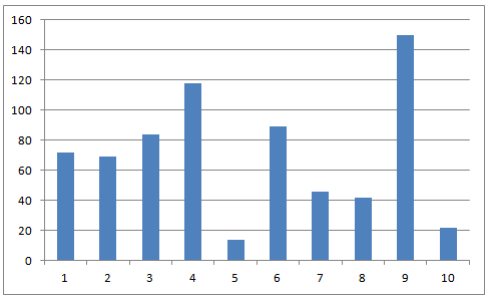
Histograma 3 10 intervalos de 36 graus. O primeiro intervalo é delimitado entre 24º e 59, o segundo por 60 e 95º e assim sucessivamente até ao último intervalo que engloba

os ângulos entre 348º e 23º.

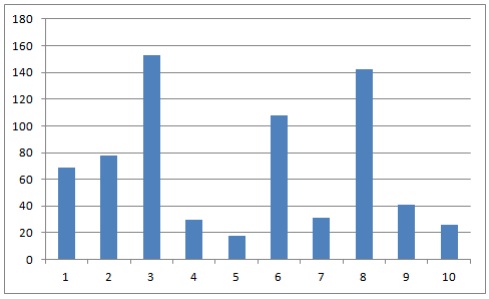
Os 3 histogramas são obtidos usando o algoritmo descrito em 5.8. Após obtidos os 3 histogramas, estes são submetidos a um filtro que remove, dos vários histogramas, os valores nulos. Ou seja, todos os intervalos que obtêm 0 no histograma são removidos. Após a filtragem, é executado o algoritmo 5.9 que permite obter o número de picos ou máximos locais para cada um dos histogramas. Este algoritmo é executado 3 vezes, uma para cada histograma. Após a terceira execução, o número de picos que identificará a sequência é o máximo obtido após a avaliação dos picos dos 3 vetores. Através do método que engloba os algoritmos 5.8 e 5.9 é possível determinar se a área encontrada na sequência é circular ou não. Foi estimado que uma área é circular se forem encontrados pelo menos 8 picos num total de 10 possíveis mudanças de direção. Na figura 5.7 estão representados 3 histogramas que correspondem aos 3 tipos de histogramas descritos em cima. A sequência base dos histogramas é a que está representada na figura 5.5. A barra vertical representa o número de vezes que foram contabilizados ângulos no intervalo representado na barra horizontal.



(a) Histograma 1



(b) Histograma 2



(c) Histograma 3

Figura 5.7: Exemplo de 3 histogramas para a parcela 5.5

O algoritmo 5.8 itera sobre os vários *trackpoints* e depois de obter o valor final da posição a incrementar, dado pela variável *finalValue*, o vetor final é incrementado em 1 nessa posição. Depois de ser devolvido após retornar, o vetor é submetido a uma filtragem, com o objetivo de remover todos os intervalos que não obtiveram contabilizações. Por exemplo, se existir um intervalo que engloba os ângulos compreendidos entre 0º e 35º e não tiver ocorrido nenhum *finalValue* nesta posição, como o resultado será 0, o intervalo

é removido do vetor final. O algoritmo é executado 3 vezes, uma para cada sub-divisão.

Listing 5.8: Algoritmo de criação de histograma

```
1
2  Input: List of <Trackpoint> (originalSequence), SD
3  Output: Array <Trackpoint> (resultList)
4
5  sizeOfClasses := 36
6
7  //iterates over the list of trackpoints
8  for i:=0 to originalSequence.size-1 do
9      create coordinate1 and coordinate2
10
11     //takes 2 coordinates
12     coordinate1 := originalSequence.get(i).middleCoordinate
13     coordinate2 := originalSequence.get(i+1).middleCoordinate
14
15     //it calculates the bearing between the 2 coordinates and increases the
           final array in the position[finalValue]
16     bearing := calculateBearing between coordinate1 and coordinate2
17     finalValue := Math.round(( (360 + bearing - i \times ( sizeOfClasses / SD ) )
           \% 360 ) / sizeOfClasses);
18
19     resultList[finalValue] :+=1
20
21 end
22
23 return resultList
```

O algoritmo que permite obter o número de picos ou máximos locais para um determinado vetor, é o que está descrito em 5.9. O algoritmo recebe como parâmetro um vetor de histograma e começa por calcular a média e o desvio padrão do histograma. A média é usada como fator de decisão para determinar se estamos num máximo local ou não. Ou seja, se o valor para uma determinada posição no histograma for superior à média, é identificado como um máximo local. O algoritmo começa por percorrer o histograma até encontrar o primeiro valor abaixo da média. Será a partir do índice em que se encontra esse valor que serão calculados os picos. Cada vez que se encontrar um valor acima de *marginValue* é incrementado em 1 o número de picos.

Listing 5.9: Algoritmo que determina o número de picos de um histograma

```
1
2  Input: List of <Trackpoint> (originalSequence), meanValueOfHistogram
3  Output: Integer numberOfSpikes
4
5  marginValue := meanValueOfHistogram
6  firstIndeBelowMarginValue := -1
7
8  //gets the first value below the marginValue
9  for i:=0 to originalSequence.size-1 do
10     if originalSequence.get(i) < marginValue then
11       firstIndeBelowMarginValue :=i
12       break
13     end
14 end
15
16 numberOfSpikes :=0
17
18 //iterates over the values in the histogram
19 for i:=firstIndeBelowMarginValue to originalSequence.size-1 do
20   value := originalSequence.get(i)
21
22   //if the value found is bigger then the marginValue it increases the number of
         spikes
23   if value >= marginValue then
24     numberOfSpikes :+=1
25   end
26
27 end
28
29 return numberOfSpikes
```

No final, após a terceira execução, é atribuído à sequência o seu número de picos, que corresponde ao número máximo de picos encontrado após avaliar os 3 histogramas. No caso da figura 5.7, a média para os 3 histogramas foi de 71, resultando no número de picos para o primeiro histograma 4, para o segundo 4 e para o terceiro 4. O resultado final foi de 4 máximos locais para o rumo traçado na figura 5.5.

Apesar de ter resultado em alguns casos, nem sempre foi possível determinar se uma sequência tem a forma circular através deste método, talvez devido a questões de parametrização. Desta forma tentou-se outra abordagem.

A segunda abordagem faz uso das funções do *PostGIS ST_Area*, *ST_ConvexHull*, *ST_ConcaveHull* e *ST_MinimumBoundingCircle*, tendo como objetivo detetar se uma determinada sequência é ou não circular. Na figura 5.8 está representada uma sequência circular delimitada pelos pontos amarelos que são os vários *trackpoints*. Dado tratar-se de um círculo praticamente perfeito, foi uma das sequências para testar a identificação da forma circular.

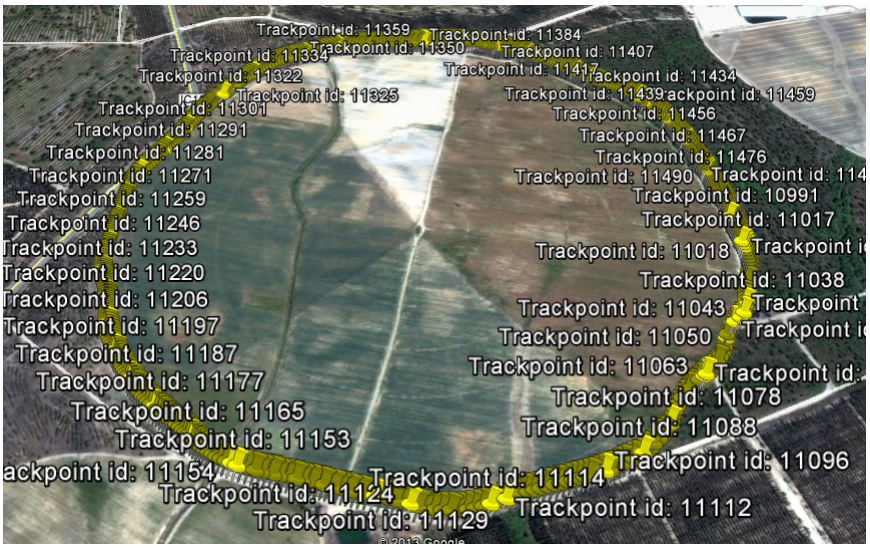


Figura 5.8: Sequência circular detetada

Foram realizadas 3 abordagens:

1. Comparar a área da sequência inicial de *trackpoints* obtida, com a área obtida pela aproximação das coordenadas da sequência inicial à forma de um círculo.

2. Submeter as coordenadas da sequência inicial de *trackpoints* a uma geometria convexa e comparar a área resultante, com a área obtida pela aproximação das coordenadas da sequência inicial à forma de um círculo.

3. Submeter as coordenadas da sequência inicial de *trackpoints* a uma geometria côncava e comparar a área resultante, com a área obtida pela aproximação das coordenadas da sequência inicial à forma de um círculo.

No ponto 1, é utilizada a função *ST_MinimumBoundingCircle* que recebe como *input* um conjunto de coordenadas e retorna um polígono que representa a aproximação dessas coordenadas à menor geometria circular onde é possível incluir com aquelas coordenadas. Ou seja, foi aplicada a função ao conjunto inicial de coordenadas da sequência. Posteriormente, aplicando a função *ST_Area*, que recebe um conjunto de coordenadas e retorna a área do polígono delimitado por essas coordenadas, ao círculo e à sequência original, não se obteve uma área aproximadamente igual. Caso se verificasse uma igualdade entre as duas áreas ou um valor muito aproximado, considerar-se-ia que a sequência teria uma forma circular. Como tal não aconteceu, o ponto 1 falhou como tentativa para a identificação da forma circular.

No ponto 2, a sequência inicial é submetida à função *ST_ConvexHull* que, sendo-lhe fornecida um conjunto de coordenadas, retorna um polígono com a forma convexa que mais se adapta às coordenadas fornecidas. Após comparar a área da forma convexa resultante, com a área do círculo obtida através da aplicação da função *ST_MinimumBoundingCircle*

às coordenadas da sequência inicial, os resultados obtidos voltaram a indicar que não se tratava de um círculo. As áreas voltaram a retornar um valor muito diferente: 390000 metros quadrados contra 500000. Na figura 5.9 é possível observar o resultado da aplicação da função *ST_ConvexHull* aos dados iniciais. É possível observar a semelhança existente com a figura 5.8. No entanto, não foi possível afirmar que se trata de uma forma circular.

Figura 5.9: Sequência circular resultante da aplicação da função *ST_ConvexHull*

O último ponto descreve a última abordagem efetuada. Segundo a documentação do *PostGIS*, a função *ST_ConcaveHull*, que tenta aproximar o conjunto de coordenadas que lhe é fornecido ao menor polígono côncavo que as engloba, obtém um melhor resultado que a função *ST_ConvexHull*. Esta função recebe para além das coordenadas, um parâmetro que delimita a percentagem dos pontos que se deseja que sejam incluídos no polígono final. Ou seja, se o parâmetro for 1, o resultado será o mesmo do *ST_ConvexHull*. O valor escolhido foi de 0.99. Mais uma vez, foi comparada a área do polígono circular criado com as coordenadas da sequência original, com a área do *ST_ConcaveHull* e os resultados voltaram a indicar não se tratar de um círculo. Houve mais uma vez uma discrepância entre as áreas, contudo, observando a figura 5.10, que representa o polígono côncavo obtido, é possível ver que há poucas diferenças entre o polígono côncavo e o original representado na figura 5.8.

Figura 5.10: Sequência circular resultante da aplicação da função *ST_ConcaveHull*

De forma a tentar perceber melhor qual o ponto de falha, foi analisado o resultado da figura 5.11. É possível observar que quando é requisitado o mínimo círculo que engloba os pontos da sequência original, representada em 5.8, o resultado aproxima-se de uma elipse. Chegou-se à conclusão que o problema é que devido ao facto de existirem mais pontos na parte superior esquerda do polígono representado, o centro de massa não se encontra no centro da figura. Assim, quando é calculada a distância média do centroide aos pontos que determinam a sequência inicial, o valor do raio obtido é superior em 50 metros ao esperado, o que resulta na diferença entre as áreas.

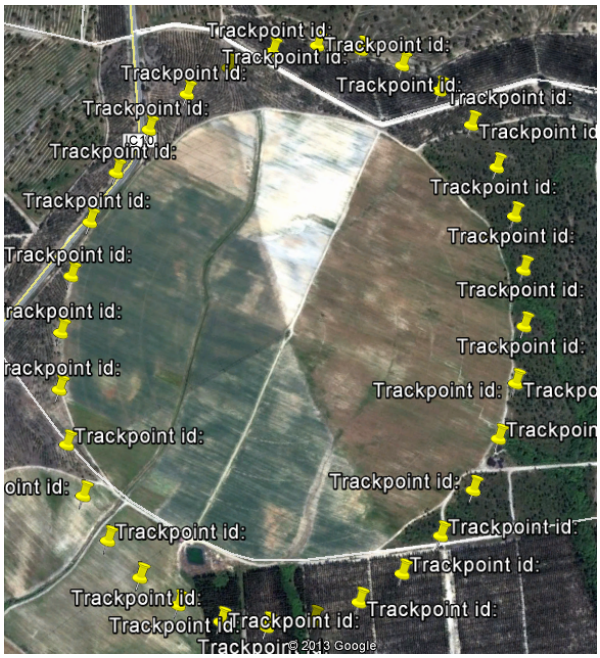


Figura 5.11: Polígono obtido pela aplicação da função *ST_MinimumBoundingCircle* à sequência original

A decisão em afirmar que uma parcela é ou não circular através da metodologia apresentada recorrendo às funções do *PostGIS ST_Area*, *ST_ConvexHull*, *ST_ConcaveHull* e *ST_MinimumBoundingCircle*, foi testada também na parcela presente na figura 5.5. Como era de esperar, a diferença de áreas entre a sequência original e as 3 abordagens foi bastante notória. A área original da parcela é 40000 metros quadrados e o resultado obtido foi de 94000 metros quadrados.

5.4.3.3 Identificação de pivôs

A identificação de pivôs, embora tenha funcionado algumas vezes, não se pode afirmar com convicção que funciona na maioria dos casos. Os pivôs podem ser circulares ou retangulares. No caso dos pivôs circulares, após a marcação da parcela, o técnico deve deslocar-se ao centro de rotação do pivô, parar, e voltar ao sítio onde terminou a marcação da parcela. Por outro lado, para marcar um pivô retangular, o técnico, depois de marcar a parcela, deve deslocar-se à outra ponta da parcela, parar e voltar ao sítio onde terminou a marcação. A identificação das marcações dos pivôs é feita nas sequências resultantes do algoritmo 5.6 e também usando o número de picos resultante da aplicação da técnica dos histogramas às sequências em causa.

A primeira pré-condição é que a sequência anterior à que está a ser analisada, para possivelmente inferir uma atividade de marcação de pivôs, tenha sido identificada como uma atividade de "Marcar parcela". Verificada a primeira pré-condição, é analisado o número de picos da sequência atual e verifica-se se houve uma movimentação até ao centro da parcela marcada na sequência anterior. Em primeiro lugar, o número de picos deve ser inferior a 4. O valor deriva do facto de o utilizador se movimentar em linha reta tanto para o centro como para a outra ponta da parcela marcada na sequência anterior à atual. Por último, é detetado se existe, pelo menos, uma coordenada que se situe no centro da sequência anterior. Por exemplo, se o técnico marcou um pivô circular, então garantidamente que existem um ou mais pontos, na sequência em que está a ser identificado o pivô circular, que estão muito próximos do centro da parcela marcada na sequência anterior à atual. No final, caso se detete a movimentação até ao centro, a sequência é identificada como a atividade "Marcar pivô circular", caso contrário é identificada como "Marcar pivô retangular".

5.5 Interface

Toda a informação extraída e identificada pela solução desenvolvida pode ser visualizada através de um *website* criado para a dissertação. Através deste, o técnico pode também corrigir os lugares e as atividades que estejam mal identificados/classificados. Toda a informação relacionada com dados geográficos é apresentada recorrendo ao *Google Maps* da *Google*. O *website* está dividido em 5 áreas distintas:

1. *GPX* original
2. *Staypoints*
3. Lugares
4. Atividades
5. *About*

Na figura 5.12 é possível observar não só o aspeto geral da interface desenvolvida mas também a informação que é possível consultar no separador "*GPX* Original". O separador permite visualizar o traço obtido pela junção de todos os *trackpoints* obtidos durante a captação efetuada pelo técnico. Cada separador permite aceder a uma área distinta. Cada área foi desenhada especificamente com o propósito de permitir visualizar e corrigir (caso faça sentido) o resultado pertencente ao nome do separador a que está associado.

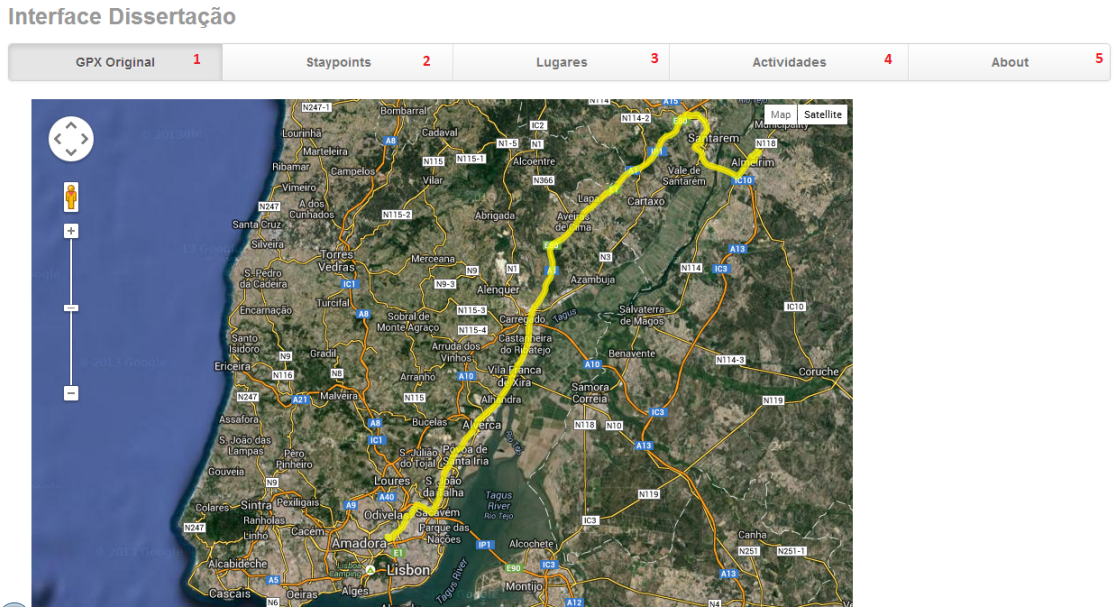


Figura 5.12: Página principal e separador "*GPX* Original".

Na figura 5.13 é possível observar a informação presente no separador "*Staypoints*". Neste separador é possível observar toda a informação relacionada com os pontos de estadia identificados pela aplicação.

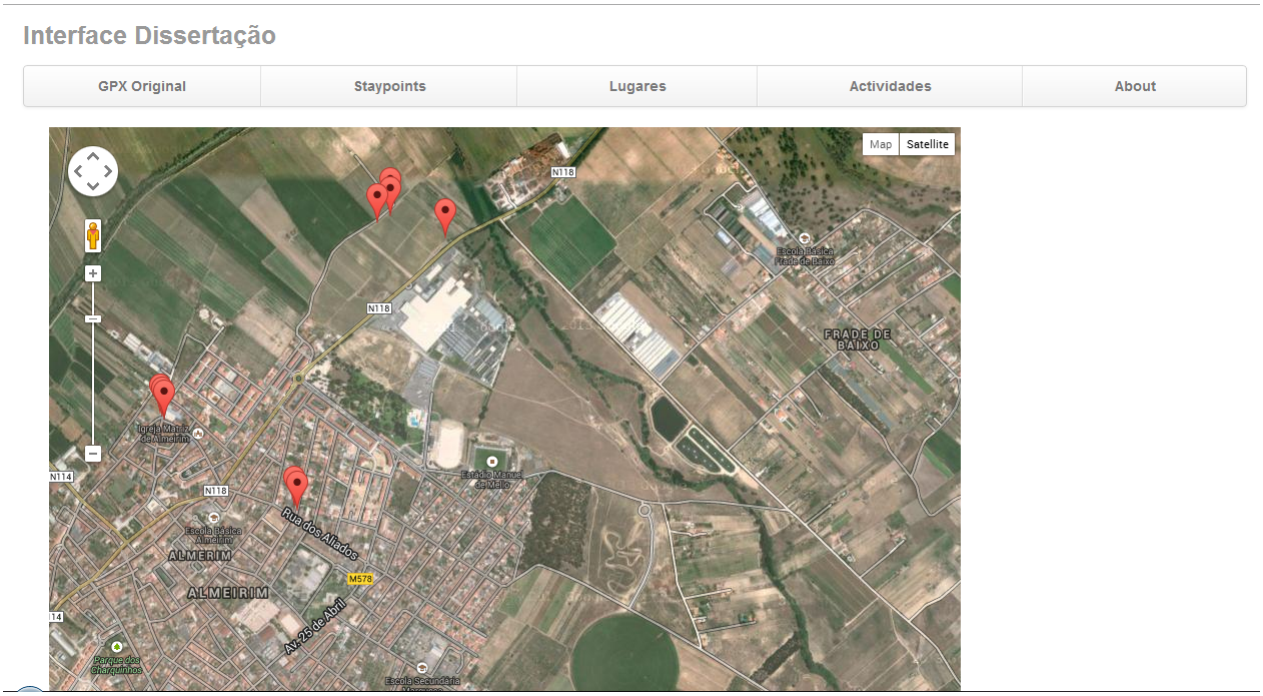
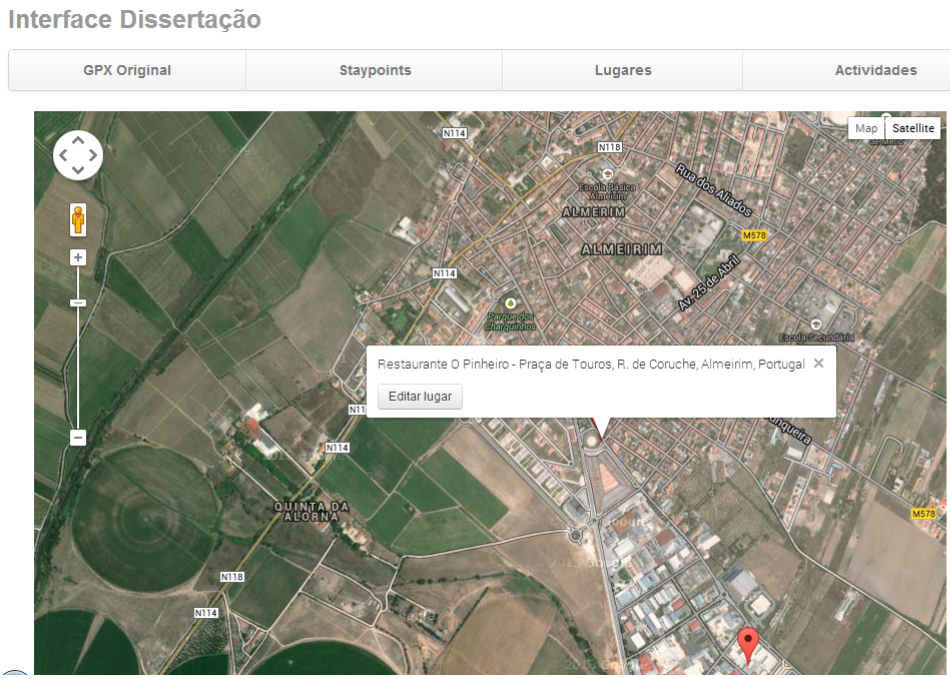


Figura 5.13: Página do separador "*Staypoints*".

A figura 5.14 apresenta o aspeto da página obtida através do separador "Lugares". Através desta página é possível visualizar todos os lugares identificados pela aplicação. A correção da informação mostrada no mapa pode ser feita através da janela presente na figura 5.15, em que é possível corrigir o nome do lugar caso este tenha sido mal classificado. Após o utilizador clicar no botão "Submeter" é realizado um pedido *REST* que regista na base de dados a alteração efetuada.

(a) Visualização geral do separador "Lugares"



(b) Janela informativa sobre lugares

Figura 5.14: Página do separador "Lugares"

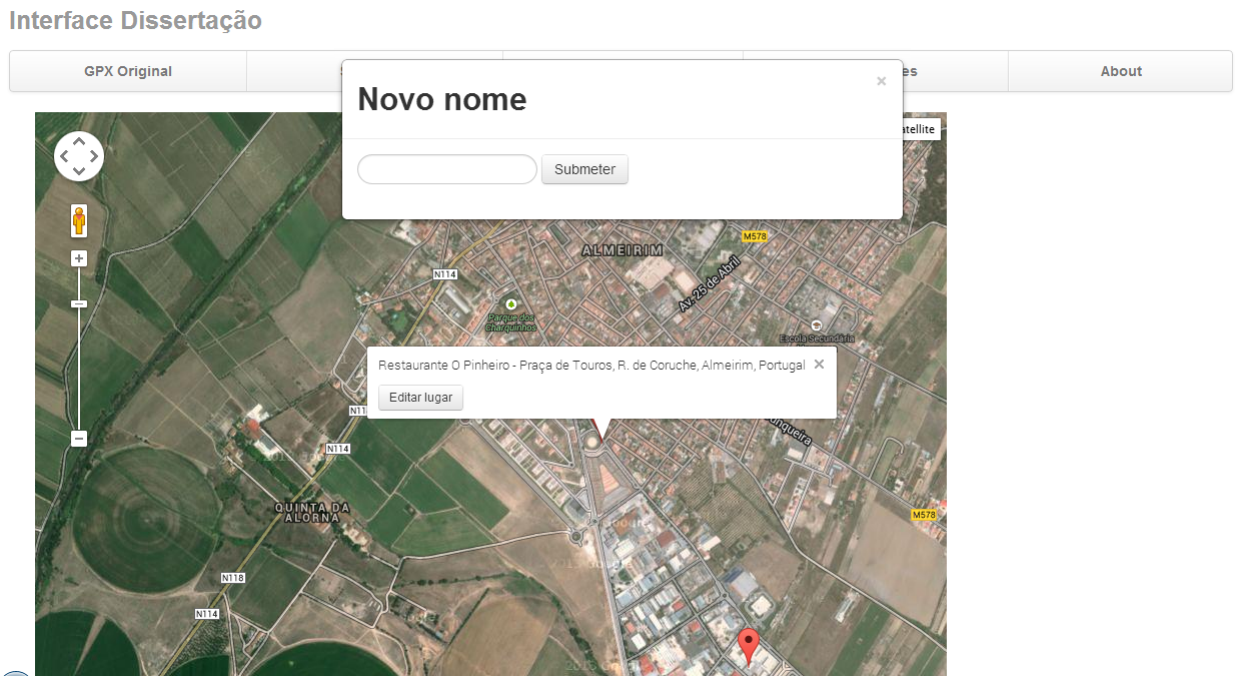


Figura 5.15: Janela de edição do nome do lugar.

A visualização de todas as atividades identificadas é realizada na janela apresentada na figura 5.16 acessível através do separador "Atividades". A barra lateral permite visualizar o troço de uma atividade ao pormenor enquanto que informação mais específica, como por exemplo a área de uma determinada parcela, está acessível clicando no troço apresentado no mapa. Tal como no separador "Lugares", existe uma forma de poder corrigir o tipo de atividade associado a um determinado troço. A correção pode ser feita através da janela de informação do troço, clicando no botão "Editar Atividade", que apresentará uma janela como exemplificado na figura 5.17.

A última área diz respeito à visualização de informações relacionadas com o autor e respetivos orientadores desta dissertação.

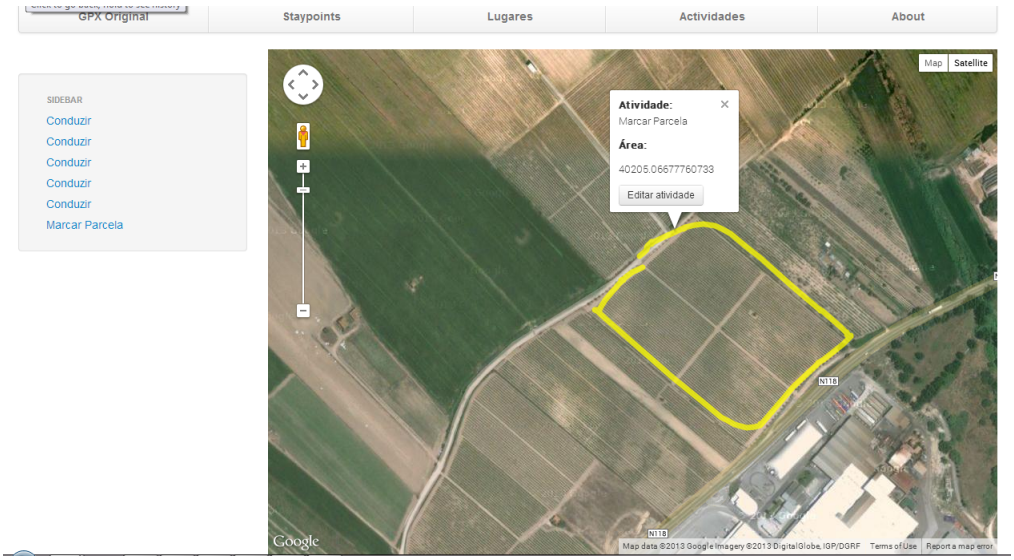


Figura 5.16: Página do separador "Atividades"

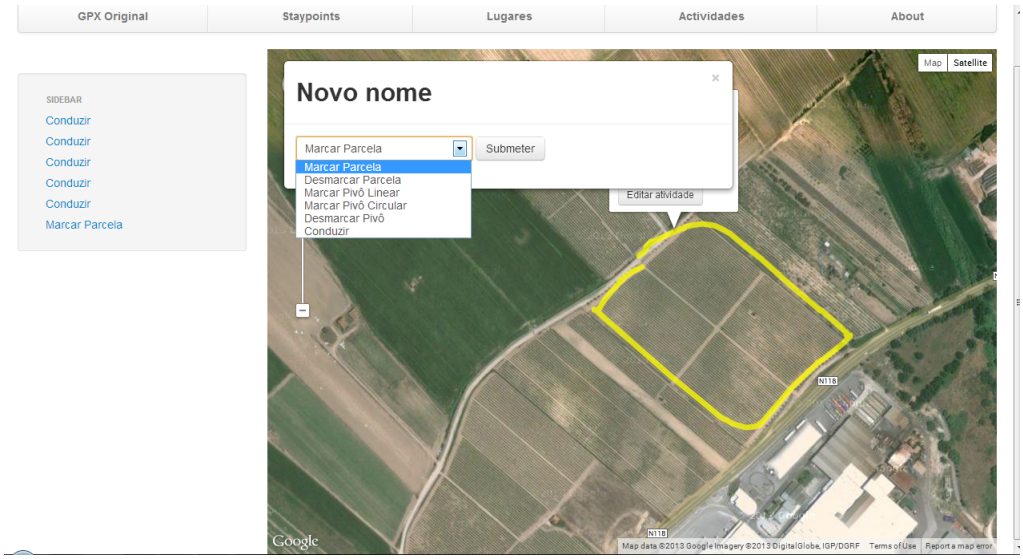


Figura 5.17: Janela de edição do tipo de atividade.

5.6 Conclusão

Ao longo deste capítulo foram apresentados todos os mecanismos e algoritmos que permitiram o desenvolvimento desta dissertação. Para desenvolver a solução criada foram utilizados alguns algoritmos que já tinham sido apresentados e descritos no capítulo 2 e justificados outros que foram desenvolvidos propositadamente para esta dissertação. Por fim, é também apresentada a interface que permite consultar os resultados obtidos quando submetida uma captação à aplicação. A avaliação dos vários resultados obtidos é detalhada no próximo capítulo.

6

Avaliação

Neste capítulo são apresentadas algumas considerações relacionadas com as razões que levaram à escolha dos valores de parametrização adotados e também as avaliações relacionadas com os resultados obtidos após a submissão de algumas captações à solução que foi apresentada ao longo do documento.

As captações nos quais são baseados os resultados foram obtidas por dois utilizadores: o Prof. Carlos Damásio e uma técnica agrícola de uma empresa relacionada com a área da agricultura. Tentou-se aproximar o mais possível o ambiente de captura à realidade, realizando as captações num ambiente não muito controlado no decorrer do dia-a-dia de ambos, sem existir a preocupação de que todos os movimentos estavam a ser guardados. A única preocupação foi a de respeitar os padrões para marcação e desmarcação de parcelas por forma a os mesmos poderem ser identificados mais tarde nos registos.

6.1 Metodologia de avaliação

A avaliação do desempenho da solução tem por base 5 captações. Os vários parâmetros que melhor se adaptam à solução foram sendo alterados e verificados, analisando os dados e os resultados das primeiras duas captações. Após obter um resultado aceitável numa determinada captação, as que já tinham sido analisadas foram submetidas novamente à aplicação por forma a verificar se não houve perca de dados. Após obter resultados estáveis, foram submetidas as outras 3 captações diretamente à solução criada com o propósito de avaliar o desempenho e os resultados com os valores dos parâmetros previamente já estabelecidos.

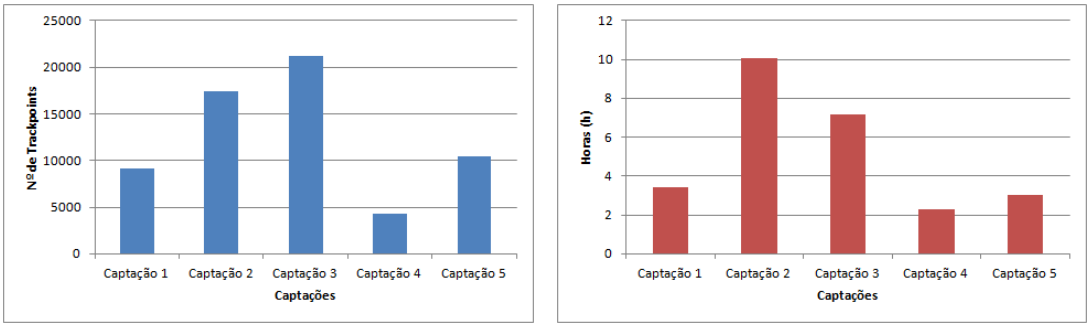
Por forma a realizar uma avaliação credível e concisa, todos os resultados obtidos

para uma determinada captação foram confirmados com a pessoa que os recolheu.

6.2 Captação de dados

A captação de dados foi efetuada usando a aplicação *OSMTracker* e após apresentar as várias captações obtidas aos coletores, a aplicação foi avaliada como uma boa escolha para a recolha de dados *GPS*. A sua avaliação, no caso desta dissertação, está de acordo com a avaliação dada na *Google play* [1], a loja de aplicações da *Google*, que foi de 4.5/5.

Na figura 6.1 estão representados 2 gráficos. O primeiro indica qual o número de *trackpoints* de cada captação enquanto que, o segundo mostra qual o tempo total que cada captação tem. É possível observar que o número de *trackpoints* tem uma relação com o tempo despendido, dado que cada *trackpoint* representa um segundo. No entanto, nem sempre a proporção é direta pois a aplicação, quando não deteta satélites suficientes para determinar a posição do utilizador ou quando este desliga o sensor *GPS*, pára de captar até voltar a obter qualidade suficiente de sinal. O tempo médio tomado pela aplicação para a extração de toda informação de atividades e lugares é aproximadamente 3 minutos. O tempo foi obtido numa máquina cujo sistema operativo é o *Windows* 7, o processador é um *Duo 7300* da *Intel* e tem de memória *RAM* 2 *GigaBytes*.



(a) Número de trackpoints em cada captação (b) Tempo dispendido em cada captação

Figura 6.1: Gráfico do número total de *trackpoints* e do tempo total de cada captação

6.3 Captações 1 e 2

6.3.1 Deteção e identificação de lugares

Os valores dos parâmetros foram decididos utilizando as primeiras duas captações e foram moldados por forma a conseguirem detetar o máximo de informação nas duas. Foram testados vários valores para os vários parâmetros de distância e tempo, sendo os valores que foram mencionados no capítulo anterior os que melhor se adaptam à solução, produzindo os melhores resultados. O mesmo se aplica à deteção de pontos de estadia.

[1] https://play.google.com/store/apps/details?id=me.guillaumin.android.osmtracker&hl=pt_PT

Todas as coordenadas de lugares identificadas pela aplicação foram consideradas corretas pelos utilizadores. No entanto, nem sempre foram identificadas como sendo o lugar real certo. A identificação do local real a que corresponde determinada coordenada foi feita de duas formas, utilizando os serviços *online* do *Google* e através de informação introduzida pelo utilizador.

Quando a aplicação tenta identificar o lugar real, tenta identificar em primeiro lugar se já há um lugar da aplicação perto da coordenada identificada, e se sim, este é classificado como igual ao que já estava na base de dados. A falta de informação de alguns lugares no início na aplicação levou a que, por exemplo, a casa do técnico não fosse classificada corretamente.

No caso da deteção através dos mecanismos do *Google*, o problema da identificação real do lugar não está associada à coordenada obtida pela aplicação, que segundo os dois utilizadores está correta, mas sim na forma como o *Google* devolve o estabelecimento mais próximo. O serviço do *Google* baseia-se num conjunto de critérios, tais como o número de pessoas que já visitou o lugar, a classificação dada pelos utilizadores ao estabelecimento ou o quão frequente é as pessoas irem aquele estabelecimento, e classifica-o como sendo mais ou menos importante. Assim, quando se procura qual o estabelecimento de determinado tipo mais perto de uma determinada coordenada, o resultado retornado é ordenado consoante a classificação do *Google*, o que faz com que por exemplo restaurantes menores não tão conhecidos, percam para os grandes. Os resultados mal classificados pela solução, podem ser corrigidos através da interface.

Na captação 1 foram encontrados 3 lugares, dos quais 1 foi identificado como mal classificado pois foi considerado como uma morada pelo *Google*, mas deveria ter sido identificado como a casa do técnico. Por outro lado, na captação 2 foram encontrados 6 lugares sendo que 2 deles foram incorretamente identificados. Um dos 6 lugares foi identificado como ruído pelo utilizador que recolheu a captação, tendo sido eliminado. O segundo lugar foi corretamente identificado mas mal classificado pelo *Google*. O restaurante resultante da procura no *Google* foi um mais conhecido perto daquele a que o utilizador foi.

6.3.2 Identificação de atividades

Toda a parametrização relacionada com a deteção de atividades e os algoritmos subjacentes à mesma foi definida tendo em conta as captações 1 e 2. Todas as atividades efetuadas pelos técnicos durante o período das captações foram encontradas e bem classificadas pela aplicação. A primeira captação continha 6 atividades enquanto que a segunda continha 8 atividades. Em ambas, a atividade mais contabilizada foi a de condução, sendo que na primeira houve 2 "Marcação de parcela" corretamente identificadas.

6.4 Captação 3

6.4.1 Deteção e identificação de lugares

Segundo os utilizadores, todos os locais foram corretamente identificados e classificados. Foram encontrados 14 lugares.

6.4.2 Identificação de atividades

No total foram encontradas 21 atividades. Após mostrados os resultados obtidos aos utilizadores, chegou-se à conclusão que 20 das atividades foram corretamente identificadas e classificadas. Contudo, uma das atividades classificada como "Marcação de pivô retangular" foi mal classificada. Na realidade era parte de uma marcação mal efetuada pela técnica.

6.5 Captação 4

A captação 4 tinha como único objetivo a deteção de uma atividade num único lugar. Todos os dados foram obtidos em cidade, como tal foi desligado um dos filtros que implica que a deteção das atividades tenha de ser feita em períodos delimitados por pontos de estadia que se encontrem no campo ou pelo menos um deles se encontre no campo. A atividade foi detetada no parque das nações, após um edifício ter sido contornado como forma de simular uma marcação de parcela.

6.5.1 Deteção e identificação de lugares

Foi encontrado 1 lugar na captação que corresponde ao lugar onde foi marcada a parcela. Devido ao ruído que existia nos dados o ponto acabou colocado no oceano, tendo sido o lugar classificado pelo técnico como mal identificado. Este problema talvez pudesse ser facilmente resolvido implementando um dos filtros descritos no estado da arte presente na secção 2.2.

6.5.2 Identificação de atividades

Foram identificadas corretamente as 3 atividades que existiam na captação:

Condução Ida para o Parque das Nações

Condução Retorno a casa do técnico

Marcação de parcela

6.6 Captação 5

6.6.1 Deteção e identificação de lugares

Foram identificados 4 lugares. Segundo o utilizador, foram corretamente identificados e classificados.

6.6.2 Identificação de atividades

Nesta captação, as únicas atividades que foram detetadas foram as de condução, não tendo sido identificada uma atividade de "Marcação de parcela". No total foram identificadas corretamente 4 atividades em 5.

Após perceber qual o motivo da não identificação da "Marcação de pacela", chegou-se à conclusão que, quando as áreas são muito estreitas, existe um problema na deteção de atividades, tanto de desmarcação como de marcação.

6.7 Conclusão

A avaliação efetuada permitiu obter algum *feedback* em relação aos vários componentes da solução desenvolvida e ao seu desempenho. Apesar do número de utilizadores e captações avaliados ser pequeno, tudo indica que a solução funcionará para outras captações. Só uma utilização em grande escala permitirá a sua avaliação por um conjunto grande e diversificado de utilizadores, que conduza a conclusões mais informadas. No entanto, a opinião dos utilizadores em relação à identificação dos vários locais e das várias atividades é positiva. O impacto que a aplicação tem na bateria do *smartphone* é um fator negativo apontado por ambos os utilizadores, devendo por isso existir uma atenção especial no seu uso que permita minorar este problema. No entanto, a solução é apontada como um contributo positivo a uma das grandes contribuições desta dissertação, a deteção de atividades agronómicas.

Os métodos e parâmetros utilizados para a deteção e identificação dos vários locais mostraram-se precisos e, na maioria dos casos, foram obtidos bons resultados. A parametrização dos vários métodos utilizados mostrou-se adequada para a solução desenvolvida. Já a identificação do estabelecimento real a partir dos métodos *online*, nem sempre demonstrou os resultados esperados.

Segundo a avaliação dos utilizadores, o método de identificação de atividades desenvolvido e implementado na solução desta dissertação, é considerado um sucesso. A solução consegue identificar grande parte das atividades que foram apresentadas no início do documento: conduzir, estar num restaurante, estar numa bomba de gasolina, detetar os vários locais visitados pelo técnico e, por fim, detetar as atividades de marcação e desmarcação de parcelas. Todavia, apesar de se conseguir identificar a área marcada ou desmarcada, nem sempre é possível determinar se a sua forma é circular. No entanto, em grande parte das vezes é possível identificar que determinada área não é circular.

7

Conclusões e trabalho futuro

O presente capítulo efetua uma análise final ao trabalho desenvolvido e aos seus componentes, apresentando as conclusões finais desta dissertação. São também apresentadas algumas propostas para algum trabalho futuro que tenha como objetivo a deteção de outras atividades do técnico agrícola e que possa complementar a solução atual.

7.1 Conclusões

Nesta dissertação é apresentada uma solução que faz uso de vários métodos que permitem extrair um conjunto de informação relacionada com a vida do dia-a-dia de um técnico agrícola. Através da recolha de captações *GPS* usando um *smartphone*, o técnico pode obter um relatório do seu dia que contém quais as várias atividades que ele praticou, bem como os lugares que visitou. O *smartphone* captura, ao segundo, todos os movimentos efetuados pelo técnico, permitindo no final, extrair e identificar toda a atividade do mesmo ao longo do período de captação. O objetivo desta dissertação foi criar uma solução que, utilizando os dados contidos nas várias captações, permite extrair um relatório completo da atividade do técnico agrícola, durante o seu dia de trabalho.

O trabalho da dissertação foi iniciado com a revisão do trabalho já desenvolvido por terceiros na área da deteção de lugares tendo por base captações *GPS*. Após o estudo dos vários mecanismos disponíveis e dos resultados que apresentavam nos vários artigos, foi usada a técnica dos pontos de estadia e a técnica de *clustering*(agrupamento) baseado em densidade. É possível assim detetar quais os pontos mais importantes que marcam o dia do técnico e agrupá-los obtendo no final o lugar real visitado. Toda a informação é complementada com outra proveniente dos serviços *online* do *Google*. Devido ao facto das técnicas apresentadas no trabalho relacionado, para a deteção de atividades, serem

complexas tanto de perceber como de implementar, o problema é abordado e resolvido usando técnicas criadas propositadamente para esta dissertação. Assim, a identificação das atividades é realizada usando os vários pontos de estadia previamente obtidos, isolando as sequências que ocorrem nos períodos que são delimitados por pontos de estadia que são precedidos e sucedidos por períodos de condução. Todo o relatório é passível de ser observado na interface criada, possibilitando assim ao técnico visualizar o trajeto total que realizou, os lugares que visitou e as atividades que exerceu. É também possível na interface corrigir informação que possa ter sido mal classificada.

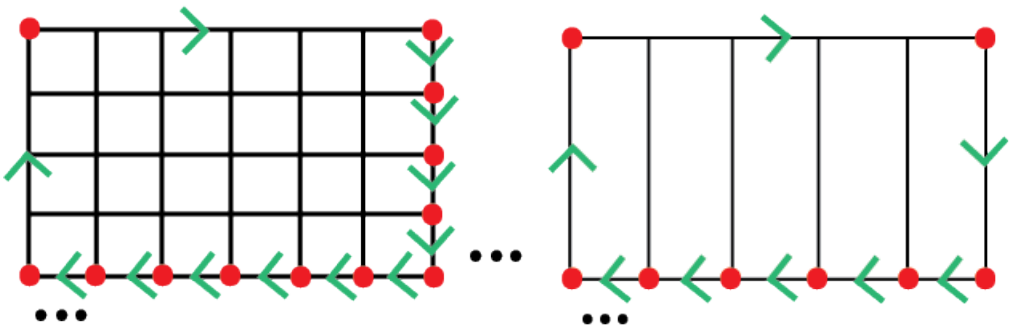
Os resultados, após a avaliação por parte dos utilizadores que recolheram as captações, são bastante bons e promissores. Apesar de o número de utilizadores, bem como o número de captações, não ser muito grande, o *feedback* é muito positivo. Segundo as avaliações obtidas, na descoberta dos vários lugares, em praticamente todas as captações são encontradas corretamente pelo menos as coordenadas dos locais, havendo por vezes algumas falhas relacionadas com os resultados do serviço do *Google*. Já a identificação das várias atividades foi um sucesso, embora ainda existam alguns problemas ocasionais. O primeiro prende-se com a identificação dos vários pivôs que nem sempre funciona como esperado. Em segundo lugar, a deteção de marcações em sequências que estejam muito próximas temporalmente ou cuja área seja muito baixa nem sempre devolve os resultados esperados. O último problema está relacionado com a deteção da forma da área identificada na marcação ou desmarcação, mais especificamente, na identificação de áreas circulares. Tal como foi explicado no capítulo da implementação, toda a teoria aponta para um sucesso da metodologia proposta, mas no final os resultados não são totalmente os esperados.

Uma característica da aplicação que foi criada para esta dissertação, é a habilidade de identificar os vários tipos de movimentos e perceber se o utilizador está, por exemplo, a conduzir. Segundo os resultados obtidos na avaliação, todas as atividades de condução exercidas pelos utilizadores foram corretamente detetadas. Após o desenvolvimento desta técnica na dissertação, foi lançado pela *Google* um complemento ao seu serviço de mapas para dispositivos móveis, que permite detetar também se o utilizador se encontra a conduzir, a andar ou se está parado. Também a *Apple*, no seu novo equipamento *iPhone 5S* integrou um *chip* que permite identificar o tipo de movimento. Isto demonstra a atualidade da investigação que é realizada nesta dissertação e a sua importância, assim como o interesse dos resultados obtidos.

7.2 Trabalho futuro

Apesar do sucesso da aplicação final desta dissertação, existem melhorias que podem ser realizadas como trabalho futuro, que podem trazer um contributo positivo à aplicação criada e à área da deteção de atividades do técnico agrónomo.

A primeira melhoria a realizar está relacionada com a deteção das marcações das linhas e das grelhas das parcelas, tal como introduzidas em 1.2. Para a marcação das linhas,



(a) Indicações para marcação de grelhas (b) Indicações para marcação de linhas

Figura 7.1: Tipos de marcações

a indicação a dar aos técnicos seria a mesma das marcações das várias parcelas, mas com a diferença que numa das arestas da parcela que estivessem a marcar, deveriam parar nos vértices que definem os intervalos das linhas em causa. Os intervalos poderiam representar um número de linhas a definir. No caso das marcações de grelhas, o mecanismo seria igual ao da marcação de linhas, mas com a diferença de que os técnicos deveriam parar em duas arestas perpendiculares, nos vértices que definem os compassos. Os padrões de marcação de grelhas e linhas podem ser observados na figura 7.1.

Para além das marcações das linhas e das grelhas, poderiam ser indicado aos técnicos métodos para efetuar a marcação e observação de armadilhas. O padrão a respeitar, no caso da marcação de armadilhas, teria como pré-condição que a parcela que contém as armadilhas já tivesse sido marcada previamente nesse dia. Assim, depois de marcada a parcela, o técnico poderia entrar dentro da mesma e, para marcar as armadilhas, deveria simplesmente parar algum tempo na sua localização. Por outro lado, para realizar as observações das armadilhas, a pré-condição era que o técnico já as deveria ter marcado anteriormente, dado que a principal diferença entre a monitorização das armadilhas e a sua marcação, seria que as monitorizações não seriam feitas no mesmo dia em que seria marcada a parcela. O padrão genérico de marcação de armadilhas está representado na figura 7.2.

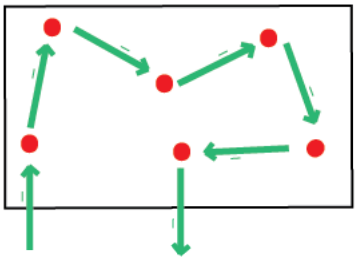


Figura 7.2: Indicações para marcação e monitorização de armadilhas

Uma última melhoria relacionada com marcações e observações está relacionada com a marcação das zonas afetadas. A marcação de uma zona afetada seria exatamente igual à da marcação de uma parcela, mas diferia no facto de que a marcação da zona afetada seria feita num dia diferente do da marcação da parcela.

Dada a instabilidade na deteção dos pivôs e na identificação das zonas circulares, uma melhoria para trabalho futuro seria a otimização e correção do processo de identificação tanto dos pivôs como das áreas circulares.

Por último, poderia ser realizada uma melhoria na interface por forma a permitir ao técnico imprimir o relatório com a informação que lhe é apresentada no ecrã, melhorar a quantidade de informação que lhe é mostrada na aplicação *Web* e permitir a correção dos limites das várias parcelas na mesma.

Bibliografia

[1] L. Aimin, F. Zhiming, X. Liming et al., "The modern precision agriculture and technological system.", *Journal of China Agricultural University*, vol. 5, nº 2, pp. 20–25, 2000.

[2] D. Ashbrook e T. Starner, "Using gps to learn significant locations and predict movement across multiple users", *Personal and Ubiquitous Computing*, vol. 7, nº 5, pp. 275–286, 2003.

[3] K. Axel, "Location-based services: fundamentals and operation", *John Wiely & Sons*, 2005.

[4] S. Bello, "Extração de informação de padrões pessoais de tempo e espaço", tese de mestrado, Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa, 2011.

[5] M. Ester, H. Kriegel, J. Sander e X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", em *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, AAAI Press, vol. 1996, 1996, pp. 226–231.

[6] W. Gilks, S. Richardson e D. Spiegelhalter, *Markov Chain Monte Carlo in practice: interdisciplinary statistics*. Chapman & Hall/CRC, 1995, vol. 2.

[7] J. Hartigan e M. Wong, "Algorithm as 136: a k-means clustering algorithm", *Applied Statistics*, vol. 28, nº 1, pp. 100–108, 1979.

[8] J. Lafferty, A. McCallum e F. Pereira, "Conditional random fields: probabilistic models for segmenting and labeling sequence data", em *Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.

[9] Y. Lee e S. Cho, "Human activity inference using hierarchical bayesian network in mobile contexts", em *Neural Information Processing*, Springer, 2011, pp. 38–45.

[10] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu e W. Ma, "Mining user similarity based on location history", em *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, ACM, 2008, p. 34.

[11] L. Liao, D. Fox e H. Kautz, "Location-based activity recognition using relational markov networks", em *Proceedings of the 19th international joint conference on Artificial intelligence*, Morgan Kaufmann Publishers Inc., 2005, pp. 773–778.

[12] ——, "Extracting places and activities from gps traces using hierarchical conditional random fields", *The International Journal of Robotics Research*, vol. 26, nº 1, pp. 119–134, 2007.

[13] J. Martin, J. Krösche e S. Boll, "Dynamic gps-position correction for mobile pedestrian navigation and orientation", em *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication 2006 (WPNC'06)*, 2006.

[14] J. Pearl, *Causality: models, reasoning and inference*. Cambridge Univ Press, 2000, vol. 29, cap. 1.

[15] A. Pestana, "Elementos de Geodesia", 2008. endereço: http://topografiasig.isep.ipp.pt/apontamentos/Apontamentos%20SIG/Textos/geodesia%20v213.pdf.

[16] J. Snively. (abr. de 2012). Gps accuracy - how accurate is it? Acedido em: 31.01.13, endereço: http://www.maps-gps-info.com/gps-accuracy.html.

[17] B. Taskar, P. Abbeel e D. Koller, "Discriminative probabilistic models for relational data", em *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 2002, pp. 485–492.

[18] M. Worboys e M. Duckham, *GIS: A computing perspective*. CRC, 2004.

[19] Y. Ye, Y. Zheng, Y. Chen, J. Feng e X. Xie, "Mining individual life pattern based on location history", em *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, IEEE, 2009, pp. 1–10.

[20] L. Yong, L. Xiushan, Z. Degui e L. Fu, "The main content, technical support and enforcement strategy of digital agriculture", *Geo-Spatial Information Science*, vol. 5, nº 1, pp. 68–73, 2002.

[21] Y. Zheng, L. Zhang, X. Xie e W. Ma, "Mining interesting locations and travel sequences from gps trajectories", em *Proceedings of the 18th international conference on World wide web*, ACM, 2009, pp. 791–800.

[22] C. Zhou, N. Bhatnagar, S. Shekhar e L. Terveen, "Mining personally important places from gps tracks", em *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, IEEE, 2007, pp. 517–526.

[23] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar e L. Terveen, "Discovering personal gazetteers: an interactive clustering approach", em *Geographic Information Systems: Proceedings of the 12 th annual ACM international workshop on Geographic information systems*, vol. 12, 2004, pp. 266–273.